8-2013

The assignment problem with dependent costs.

Ghazal Tariri
University of Louisville

THE ASSIGNMENT PROBLEM WITH DEPENDENT COSTS

By

Ghazal Tariri

B.S., Sharif University of Technology, Tehran, Iran, 2007

M.S., Isfahan University of Technology, Isfahan, Iran, 2009

A Dissertation

Submitted to the Faculty of the

J. B. Speed School of the University of Louisville

in Partial Fulfillment of the Requirements for

the Degree of

Doctor of Philosophy

Department of Industrial Engineering

University of Louisville

Louisville, Kentucky

August 2013

THE ASSIGNMENT PROBLEM WITH DEPENDENT COSTS

By

Ghazal Tariri

B.S., Sharif University of Technology, Tehran, Iran, 2007

M.S., Isfahan University of Technology, Isfahan, Iran, 2009

A Dissertation Approved on

July 25, 2013

by the following Dissertation Committee:

_____

Dr. Gail W. DePuy, Dissertation Director

_____

Dr. William Biles

_____

Dr. C. Tim Hardin

_____

Dr. Ming Ouyang

DEDICATION

This dissertation is dedicated to the bright memory of my mother,

Ms. Fakhrieh Sabeh

Who left us soon but she was always there for me when I needed encouragement, motivation, and strength to continue with my education

And my wonderful father,

Mr. Mohammad Ali Tariri

For his love and support

ACKNOWLEDGMENTS

The author wishes to express her gratitude to her advisor, Dr. DePuy, who was abundantly helpful and offered invaluable assistance, support, and guidance. Her guidance and inspiration along the way were so helpful and made this dissertation possible. Her understanding and consideration also made working with her enjoyable. Deepest gratitude is also due to the members of my dissertation committee, Dr. Biles, Dr. Hardin, and Dr. Ouyang. They have generously given their time and expertise to better my work.

The author wishes to express her love and gratitude to her beloved family; and her friends for their understanding and endless love, through the duration of her studies.

ABSTRACT

THE ASSIGNMENT PROBLEM WITH DEPENDENT COSTS

Ghazal Tariri

July 25, 2013

Assigning workers, each with their own skill set, to tasks which demand different skills in an efficient manner is a challenging problem that often requires workers to receive additional training. The training of workers is very costly with Training Magazine's Annual Industry Report stating 58.5 billion dollars were spent in 2007 on employee training in the United States. Therefore assigning workers to tasks in such a way as to minimize the overall training costs is an important problem in many organizations.

In this research, the assignment problem with dependent cost is considered, i.e. the training cost associated with assigning a worker to a particular task depends on the training the worker receives for their other assigned tasks. Once a worker is trained in a skill that training will available for any additional tasks that may be assigned. The problem is formulated mathematically as an integer linear program. Based on past research, high quality solutions to large-size problems are difficult to obtain. This research develops and upper bound approach and three heuristic solution methodologies. The basic idea of the heuristics is to form groups of tasks which require similar skills,

then assign a worker to the task group. The Shortest Augmenting Path (SAP) algorithm of Jonker and Volgenant is known to quickly find the optimal assignment of N workers to N tasks. This SAP algorithm will be used in this research after grouping the tasks into N groups which can then be assigned to the N workers. The task grouping heuristic methods developed in this research were tested for several randomly generated large-sized data sets. Results showed an average 7.34% improvement compared to previous solution methods.

Additionally to consider workers' preferences, a multiple-objective model is presented for the skills management problem to maximize workers' preferences and aggregate training while minimizing training cost. The model is demonstrated for randomly generated data sets.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In every company throughout the world, management teams face many complications, including worker-task assignments. Being able to assign tasks that require varying skill levels to workers with differing abilities is critical. Worker training programs can be used to raise workers' competencies but training is often expensive. Training requires both time and money and workers must be trained in order to understand and complete their assigned task; therefore, cost will accumulate each time a worker must complete training. As reported by Training magazine's annual Industry Report, 58.5 billion dollars were spent in 2007 on employee training in the United States. Accordingly, assignments should be made so that the required training and the total cost are minimized.

Additional costs of delayed work and reduced quality can be incurred if poor task to worker assignments are made. When workers are not properly assigned tasks, it is possible the worker will not have the proper skills to complete the task and therefore, cost the company money. It is also possible, because of poor worker-task assignments, that qualified workers are overwhelmed with too many tasks and not enough time to complete them.

Turn-over is another issue that arises. Worker turn-over and product turn-over are both problems companies face. Processes must be adjusted each time new workers have replaced old workers. This also applies for when new tasks (products) are introduced and or the company makes changes to the current task. These issues of extra expenses and poor quality work motivate the need for research in this worker-task assignment or skills management area.

It is critical to examine methods that assist with properly assigning tasks to the workers to the proposed models. Previously, assignments like these where accomplished manually by management teams. This method is legitimate as long as the criteria are met. The problem with this approach is that the task assignment becomes more complex when the number of workers and the number of tasks increase. To find high quality solutions in a reasonable amount of time becomes increasingly difficult. Therefore, it is important to develop an automated algorithm that finds optimal solutions or near optimal solutions for large-scale problems.

1.1.Problem Statement and Contribution

Because workers retain their training, the training cost, in terms of both time and money, are dependent on which tasks have been assigned to a worker and therefore, what training the worker receives. Different tasks can include the same skill and once a worker receives training in a skill then that training could be used for several tasks. This is the idea of dependent costs and will be considered in this research. This skills management problem with dependent cost is mathematically introduced by Depuy *et al* (2006).

When defining the skills management problem, there are three terms that need to be defined; task, skill and level. A task is a specific job to be completed. Skills are a set of capabilities a worker must have to complete a task. Skills are also are known as a competency framework or skills matrix. Each task requires a set of skills and multiple tasks can require the same skills. Different workers will posses different levels of expertise in these skills. The skill levels can be simply defined such as novice, intermediate, and expert. Therefore, a task is a set of skills at certain levels. Workers can receive training to increase their skill levels but training can require time and money. Clearly a worker's initial skill level will affect the cost associated with increasing the level. Once a worker receives training in a particular skill, their skill level is increased and the worker is capable of performing that skill at the trained level for any task requiring that skill. The most useful skills management is an ongoing process where workers assess and update their recorded skill sets regularly. A generalized example of task can be to change a flat tire, needed skills for this task would include proficiency in finding the required equipment, understanding of the use of a jack, ability to loosen lug nuts with a lug wrench, knowledge of undercarriage of car, and talent for fitting the spare tire within the wheel well. Some cars may have more complex jacks or very tight lug nuts. These issues would require more advanced skills than others.

Previous works related to assignment problem mostly focus on methods to solve general assignment problems and the possibility for the transfer of skills from one task to another is never entertained. In other words, previous models have ignored the potential to cut down on any further training needed for additional tasks that may be assigned. Depuy *et al* (2006) obtained optimal solutions for small skills management problems with

dependent cost using LINGO software. Depuy *et al* (2008) and Jackson *et al* (2008) presented several heuristics algorithms to solve the skills management problem. However, the optimal solution for larger problems still cannot be found in a reasonable time, thereby motivating the development of the solution heuristics developed in this research. An upper bound and three heuristic methods are developed to assign tasks to workers in this dissertation. The *k*-means clustering method is used to classify tasks based on their required skills to decrease the problem complexity. The methods are tested on several data sets used in previous research and the results are compared.

Additionally, a revised and extended skills management model is investigated to include considerations for worker preferences. The problem is cast as a multiple-objective optimization problem (MOOP), which seeks to minimize training cost while maximizing aggregate training and maximizing worker satisfaction.

1.2. Outline

The remainder of this dissertation is organized as follows:

In Chapter 2 a comprehensive literature review of assignments models and relevant details to this application is presented. Additionally, previous solution methodologies for the assignment problem with dependent costs are cited. In Chapter 3, an upper bound and two solution algorithms are developed and a data clustering method is applied. Chapter 4 includes the results of these methodologies and a comparison to previous methods. Chapter 5 demonstrates how an extended model may better explain real world applications than previous models. Finally, Chapter 6 presents the conclusions and future study plans for this research study.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

This chapter presents two main sections, literature and definitions relevant to the generalized assignment problem as well as the skills management problem and solution techniques for solving these two types of problems. Also, applications of these problems are discussed.

2.2. The Generalized Assignment problem

The assignment problem is one of the first studied classical combinatorial optimization problems and its history started with the work of G.Monge (1784), albeit concealed as a continuous problem, and often called a transportation problem. The problem was formulated in modern way by a psychologist, R. L. Thorndike (1950). In its most general form, the problem has a number of agents and tasks. Assigning each task to agents causes some cost. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized.

The classical generalized assignment problem (GAP) is a well-known, NP-complete combinatorial optimization problem stated as finding a minimum-cost assignment of tasks to workers in which multiple assignments of tasks to workers are limited by some resource (Feltl 2003). It seems the first referred GAP in literature as a 0-1 special case was by Kuhn (1955).

GAP has been applied in many real world problems ranging from jobs assigned to computer networks (Balachandran 1972) to machine loading in flexible manufacturing systems (Mazolla *et al.*, 1989.) to facility location (Ross and Soland 1977). Applications referenced include vehicle routing, fixed charge location problems, scheduling projects, allocating storage space, designing communication networks, scheduling payments on accounts, assigning software development tasks to programmers, scheduling variable length TV commercials, and assigning ships to overhaul facilities (Pentico 2007).

2.2.1. Model Formulation for GAP

The GAP can be formulated as an Integer Linear Programming (ILP) model with binary variables. The GAP model presented by J. K.Karlof (2005) is shown below.

Parameters:

n= number of workers i= {1,2,…,n}

m= number of tasks j= {1,2,…,m}

C_{ij}=cost of task j being assigned to worker i

R_{ij}=amount of resource required for task j by worker i

B_i=resource units available to worker i

Decision Variables:

$$X_{ij} = \begin{cases} 1, & worker\ i\ is\ assigned\ to\ task\ j \\ 0, & otherwise \end{cases}$$

The 0-1 ILP model is:

$$\text{Minimize } z=\sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} \qquad (2\text{-}1)$$

Subject to:

$$\sum_{j=1}^{m} r_{ij}x_{ij} \le b_i, \qquad \forall i \qquad (2\text{-}2)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \qquad (2\text{-}3)$$

$$x_{ij} = 0 \quad or \quad 1, \qquad \forall i,j \qquad (2\text{-}4)$$

The objective function (2-1) minimizes the total cost of the assignment. Constraint (2-2) enforces resource limitations for each worker, and constraint (2-3) ensures all tasks are assigned to exactly one worker. Although general purpose ILP solvers have become more effective in solving general ILP problems, GAP still remains difficult to solve to optimality.

2.2.2. GAP Solution Methodology

The solution methodologies for the GAP problem are various. The first formal computational work addressing the GAP was by Ross and Soland (1975). Their solution methodology was Branch and Bound and it was the first published instance of Lagrangian relaxation (LGR) applied to an ILP problem. Fisher *et al.*(1981) formalized the use of Lagrangian relaxation and devised a dual multiplier adjustment procedure that tightens the initial LGR. Jornsten and Nasberg (1986) reformulated the GAP by

introducing a set of auxiliary variables and coupling constraints. Through the late 1980s the maximum size of GAP problems to be solved optimally remained at about 500 binary variables.

In the early 1990s some heuristic approaches for solving large GAP problems optimally were devised. Cattryese *et al* (1994) used a column generation/set partitioning approach to generate good feasible solutions and mixed it by Lagrangian techniques to reduce the gap between the lower (LB) and upper (UB) bounds. Wilson (1997) developed a dual-type algorithm and joined it with a search strategy to find good feasible solutions. Chu and Beasley (1997) developed a genetic algorithm-based heuristic that generates very good feasible solutions. Yagiura *et al.*(2004) devised a tabu search algorithm using an ejection chain approach to govern the neighborhood search for feasible solutions.

Another solution approach mixes heuristic and optimizing approaches. Amini *et al.* (1999) developed an efficient heuristic for smaller problems and used an extensive statistical experimental design and analysis to compare it to earlier optimizing approaches. Savelsbergh (1997) customized a general-purpose solver, MINTO, to solve GAP problems optimality using a column generation approach.

2.2.3. Modifications to GAP

Other researchers have expanded the GAP model to include other factors. Elastic GAP allows agent resource capacity to be violated at additional cost (Nauss 2004). The objective function in elastic GAP sums the costs of the assignments and the unused time and overtime costs for agents. The additional constraints for model are to enforce resource limitations and limits for unused time and overtime.

Profit maximization is another modification of GAP in that both the assignments of jobs to available capacitated resources (agents) and the degree of resource consumption associated with each assignment must be determined (Rainwater *et al.*, 2009). The objective function maximizes total revenue received when tasks are assigned to an agent at a specific level.

The special case of GAP which is considered in this research is the assignment problem with dependent cost (DePuy *et al*., 2006). In these cases, the cost of assigning a task to a worker is dependent upon which other tasks are assigned to the worker. There is no similar problem in previous literature, to date.

2.3. Skills Management Problem

The skills management problem addresses the effective utilization of employee skills in a company (Ley *et al.* 2003). There are many issues related to skills management concepts. Competence Performance Theory studied by Korossy (1997), uses mathematical structures to establish prerequisite relations on the competence and the performance level. Skills Management Information Systems (SMIS) developed by J.Hasebrook (2001), enables the user to select learning modules according to her or his individual demands, prior knowledge, and time schedule. The competency management concept was initially used in align with human resource processes to fulfill human resource constraints set and company strategy by Green (1999).

The skills management problem addressed in this research is a type of the generalized assignment problem. In an effort to retain the current workforce, each worker must be assigned at least one task and each task is assigned to only one worker. So each worker may be assigned multiple tasks. As mentioned previously, each task is comprised

of a set of skills at specific levels and each worker has their level of each skill. A supervisor can determine the levels of skill set of the workers. A skills gap occurs when a worker is assigned to task with one or more skills at a higher level than the worker posses. Then that worker must be trained to meet the required skill level. The skills management problem has a feasible solution when all workers can be trained for assigned tasks and complete them in equal or less than capacity time and/or proprietary funds.

This research investigates a skills management problem with dependent cost and, as such differs from the GAP. In this model once a worker is assigned to a task and trained to a certain level for a specific task, their skill level is updated for all additional future assigned task, i.e., once a worker is trained in a skill, that training will be transferred to any further training needed for additional tasks that may be assigned.

2.3.1. Model Formulation for Skills Management

This skills management problem can be described mathematically as originally introduced in DePuy *et al*. (2006).

The used parameters are:

S_{ik} = worker i's skill level for skill k

R_{jk} = required skill level for task j's skill k

T_j = length (# hrs) of task j

A_i = capacity (# hrs) of worker i

C_{klm} = cost associated with raising a worker's skill level on skill k from level l to level m

E_{klm} = time required (# hrs) to raise a worker's skill level on skill k from level l to level m

The decision variables are defined as:

$$X_{ij} = \begin{cases} 1, & worker\ i\ is\ assigned\ to\ task\ j \\ 0, & otherwise \end{cases}$$

$$Z_{ikS_{ik}m} = \begin{cases} 1, & worker\ i\ receives\ training\ on\ skill\ k\ to\ raise\ skill\ level\ from\ Sik\ to\ m \\ 0, & otherwise \end{cases}$$

$$W_{ik} = \begin{cases} 1, & worker\ i\ does\ not\ need\ further\ training\ in\ skill\ k \\ 0, & otherwise \end{cases}$$

The 0-1 ILP model is:

Minimize Training Cost: Minimize $\sum_i \sum_k \sum_m C_{kS_{ik}m} Z_{ikS_{ik}m}$ (2-5)

Subject to:

Determine Needed Training:

$$S_{ik}W_{ik} + \sum_{m>S_{ik}}^{5} mZ_{ikS_{ik}m} \geq R_{jk}X_{ij} \qquad \forall i,j,k \qquad (2\text{-}6)$$

$$W_{ik} + \sum_{m>S_{ik}}^{5} Z_{ikS_{ik}m} = 1 \qquad \forall i,k \qquad (2\text{-}7)$$

All tasks assigned:

$$\sum_i X_{ij} = 1 \qquad \forall j \qquad (2\text{-}8)$$

All workers assigned at least one task:

$$\sum_j X_{ij} \geq 1 \qquad \forall i \qquad (2\text{-}9)$$

Worker Capacity:

$$\sum_j T_j X_{ij} + \sum_k \sum_m E_{kS_{ik}m} Z_{ikS_{ik}m} \leq A_i \qquad \forall i \qquad (2\text{-}10)$$

Binary Variables:

$$X_{ij} \in \{0,1\}, \quad Z_{ikS_{ik}m} \in \{0,1\}, \quad W_{ik} \in \{0,1\} \qquad \forall i,j,k,m \qquad (2\text{-}11)$$

The objective function, equation (2-5), minimizes the overall training cost of the assignment. Constraints (2-6) and (2-7) determine the total training needed by a worker to meet the skill levels required for each assigned task. Constraints (2-8) ensure that all tasks have been assigned and that each task is assigned to only one worker. Constraints (2-9) specify that each worker must be assigned at least one task. The (2-10) constraints make sure the total workload assigned to a worker (i.e. task time plus training time) does not exceed the worker's capacity. Finally, constraints (2-11) define all decision variables to be binary.

2.3.2. Skills Management Solution Methodology

As formulated in the previous section, the skills management model has been solved by DePuy *et al.* (2006) using exact methods, a greedy algorithm, and a meta-heuristic approach. Exact solution methods only worked for relatively small problem sizes of 60 workers, 60 tasks, and 60 skills (DePuy *et al.* 2006). DePuy et al. (2009) presented a greedy algorithm and meta-heuristic solution approach for medium sized problems. These techniques are discussed in the following section. However, these existing solution approaches do not obtain the desired solution quality for large sized problems, i.e. hundreds of workers and tasks. This research project will develop solution techniques for these large skills management problems.

2.3.2.1. Greedy Assignment Algorithm

Generally a greedy algorithm makes the locally optimal choice at each stage in the problem with no regard for the global optimal solution. The Greedy Assignment algorithm developed by DePuy *et al.* (2008) for the skills management problem has two phases. The first phase assigns exactly one task to each worker to make sure workforce preservation is met. The second phase assigns the remaining tasks to workers with unfilled capacity. For models with fixed assignment, those fixed assignment are made before phase 1 and any worker involved with a fixed assignment will not be included in phase 1 but will be considered for additional task assignments in phase 2.

In phase 1 of the DePuy *et al.* (2008) Greedy Assignment algorithm, each worker is assigned one task. The total training cost for each worker to complete all the tasks is first calculated to determine the least skilled workers, i.e. the workers with the largest total training cost. The workers are sorted from highest to lowest total training costs (i.e. sorted from least skilled to most skilled workers) and the workers, in sorted order, are assigned their least training cost (i.e. easiest) task. Once a task is assigned, it is deleted from the task list. At the end of phase 1, each worker is assigned exactly one task. The worker capacities and skills set are updated based on these phase 1 assignments. After phase 1, phase 2 assigns all remaining tasks to workers.

In phase 2, total training cost for all workers to complete each unassigned task is calculated and the unassigned tasks are ordered from the most difficult task to the easiest task (i.e. sorted from largest to smallest total training cost). These ordered tasks are assigned to workers with each task assigned to most capable (i.e. least training cost)

available worker. Once a task is assigned, it is deleted from the list and the worker's capacity and skills set are updated. At the end of phase 2, all tasks have been assigned.

Figures 1 and 2 show the pseudocode for phases 1 and 2, respectively, using the Greedy Assignment algorithm (Figures 1 and 2 from DePuy *et al.*, 2008). The Greedy Assignment algorithm can get stuck at a local optima and therefore differ greatly from the global optimal value. The meta-heuristic, Meta-RaPS, discussed in the next section offers a way to prevent this Greedy Assignment algorithm from getting stuck in a local optimal.

```
Calculate total training cost for each worker over all tasks, total_worker_cost matrix
Do until each worker is assigned one task
    Find unassigned worker with maximum total_worker_cost, max_cost_worker
    Find unassigned task with minimum training cost for max_cost_worker, min_cost_task
    Assign min_cost_task to max_cost_worker
    Update skill set for assigned worker based on training required for assigned task
    Update total_worker_cost and total_task_cost for assigned worker and task
    Update worker_capacity for assigned worker
    Update total_training_cost
Loop
```

Figure 1. Pseudocode for Greedy Assignment algorithm Phase 1 (DePuy *et al*., 2008).

```
Calculate total training cost for each unassigned task over all workers, total_task_cost matrix
Do until all tasks are assigned
    Find unassigned task with maximum total_task_cost, max_cost_task
    Find worker with minimum training cost for max_cost_task and available worker
                Capacity, min_cost_worker
    Assign max_cost_task to min_cost_worker
    Update skill set for assigned worker based on training required for assigned task
    Update total_worker_cost and total_task_cost for assigned worker and task
    Update worker_capacity for assigned worker
    Update total_training_cost
Loop
Print total_training_cost and assignments
```

Figure 2. Pseudocode for Greedy Assignment algorithm Phase 2 (DePuy *et al*., 2008).

2.3.2.2. Meta-heuristic Solution Approach

Meta-RaPS (Meta-heuristic for Randomized Priority Search) is a general and high level strategy to include randomness in greedy construction heuristics as a way to avoid local optima (DePuy and Whitehouse, 2001; DePuy *et al.*, 2002). It integrates sampling, priority rules and randomness. Meta-RaPS has been applied to a variety of combinatorial problems such as the Set Covering Problem (Lan *et al.*, 2007), the Traveling Salesperson Problem (DePuy *et al.*, 2005), the Knapsack Problem (Moraga *et al.*, 2005), machine scheduling (Hepdogan *et al*., 2009) and the Resource Constrained Project Scheduling Problem (DePuy and Whitehouse, 2001). It has demonstrated good performance in terms of both solution quality and computation time with respect to other meta-heuristics such

as genetic algorithms, neural networks, and simulated annealing. The Meta-RaPS heuristic uses %priority and %restriction parameter to include randomness in the solution approach.

The Meta-RaPS algorithm, using the %priority parameter determines how often the assignment specified by the Greedy Assignment Algorithm is used versus when an assignment that is close to the greedy assignment will be made. The rest of the time (i.e. 100%-%priority) the assignment whose cost is within %restriction of the cost of the greedy algorithm assignment will be made. An 'available' list of those assignments whose cost is within %restriction of the cost of the greedy algorithm assignment is formed. An assignment is randomly picked from this available list. These parameters and the randomness prevent the model from getting stuck in local optimal. The values of the % priority and % randomness parameters are determined experimentally. The Meta-RaPS Greedy Assignment heuristic utilizes the Meta-RaPS concept in both phase 1 and phase 2 of the Greedy Assignment algorithm. Figures 3 and 4 show the pseudocode for phases 1 and 2, of the Meta-RaPS Greedy Assignment heuristic (Figures 3 and 4 from DePuy *et al.*, 2009).

```
Calculate total training cost for each worker over all tasks, total_worker_cost matrix
Do until each worker is assigned one task
    Find unassigned worker with maximum total_worker_cost, max_cost_worker
    Find unassigned task with minimum training cost for max_cost_worker, min_cost_task
    P = RND (1, 100)
    If P ≤ %priority Then
        Assign min_cost_task to max_cost_worker
    Else
        Form available list of unassigned workers whose total_worker_cost is within
        %restriction of maximum total_worker_cost and that worker's associated unassigned
        tasks within %restriction of min_cost_task
        Randomly choose worker/task pair from available list and make assignment
    End If
    Update skill set for assigned worker based on training required for assigned task
    Update total_worker_cost and total_task_cost for assigned worker and task
    Update worker_capacity for assigned worker
    Update total_training_cost
Loop
```

Figure 3. Pseudocode for Meta-RaPS Greedy assignment Heuristic Phase 1 (DePuy *et al.*, 2009).

```
Calculate total training cost for each task over all workers, total_task_cost matrix
Do until all tasks are assigned
    Find unassigned task with maximum total_task_cost, max_cost_task
    Find worker with minimum training cost for max_cost_task and available worker
                Capacity, min_cost_worker
    P = RND (1, 100)
    If P ≤ %priority Then
        Assign max_cost_task to min_cost_worker
    Else
        Form available list of unassigned tasks whose total_task_cost is within %restriction
        of maximum total_task_cost and that task's associated unassigned workers within
        %restriction of min_cost_worker
        Randomly choose worker/task pair from available list and make assignment
    End If
    Update skill set for assigned worker based on training required for assigned task
    Update total_worker_cost and total_task_cost for assigned worker and task
    Update worker_capacity for assigned worker
    Update total_training_cost
Loop
Print total_training_cost and assignments
```

Figure 4. Pseudocode for Meta-RaPS Greedy assignment Heuristic Phase 2 (DePuy *et al.*, 2009).

2.3.2.3. Shortest Augmenting Path (SAP)

The Shortest Augmenting Path (SAP) algorithm provides optimal results to the generalized assignment problem (Jonker and Volgenant, 1987). The SAP algorithm has been used in several applications, such as the allocation of tasks to multifunctional workers (Corominas *et al.*, 2006) and the solution of the minimum product rate variation problem (Moreno, 2007).

SAP was developed to find the optimal solution to the classical assignment problem that n workers should been assigned to n tasks. Therefore the SAP was applied to phase 1 of the skills management solution technique where each worker is assigned one task.

The interested reader is referred to Jonker and Volgenant, (1987) and Jackson *et al.* (2008) for the specific details of SAP and SAP as applied to the skills management problem.

The SAP is useful for one-to-one assignments (where the number of tasks is equal to the number of workers). Because the number of tasks usually exceeds the number of workers, the SAP algorithm is applied in the phase 1 of assignment. However a determination of which tasks to assign in phase 1 must be made. Jackson et al. (2008) suggest an approach uisng Meta-RAPS to determine which tasks are assigned in phase 1 using SAP. Phase 2 of the previous method (as shown in figure 4) can be used for the assignment of the remaining tasks. The pseudocode for phase 1 of the Meta-RaPS SAP algorithm is shown in Figure 5 (from Jackson *et al*, 2008).

```
Do until each worker is assigned one task
    n = #workers
    Calculate total training cost for each task over all workers, total_task_cost matrix
    Sort total_task_cost from smallest to largest
    Form available list of tasks whose total_task_cost is within %restriction of the nth
    smallest total_task_cost.
    Randomly choose n tasks form available list
        Find the worker with the minimum cost, min_cost_worker, for a given task
        If min_cost_worker is unassigned
            Assign task to min_cost_worker
End If
        Form list of available workers
        Do for 2 iterations
        Choose available_worker
        Find min_cost associated with available_worker
        Recalculate total cost
        Assign best_task to available worker
        Loop
        For available workers remaining
        Find worker/task pair with minimum cost
        If related task is unassigned Then
        Go To "Augmentation Code"
        End If
        Update Cost
        Find related task and calculate "new cost"
        If 'related task' is unassigned Then
        Go To "Augmentation Code"
        End If
**Augmentation Code**
Find related task and its cost
Find worker with the shortest path value for related task
Assign task to worker
Next available worker
For all assigned workers
Update worker capacities
Update total cost
Next Worker
Update workerskill, total_worker_cost, total_task_cost,
```

Figure 5. Pseudocode for Shortest Augmenting Path Algorithm Phase 1 Only (Jackson *et al*, 2008).

2.3.3. Performance of Previous Techniques for Skills Management Problem

As mentioned previously, initially this skills management assignment problem with dependent costs was presented by DePuy *et al.* (2006). The math model was an integer program and solved using commercially available solver software, LINGO, for a relatively small sized problem. However the solver software was unable to find a feasible solution in a reasonable computation times for several hundred workers and tasks. It took over 24 hours for large size problem on an Intel Pentium 4 PC with 1.00 GB of RAM. Therefore, an alternative solution methodology needed to be developed. Some heuristics and Meta- heuristics were presented in a previous section but a good heuristic method for large problems to improve solution quality remains to be developed.

The three solution methodologies (Greedy Assignment Algorithm in both phases, Meta-RaPS SAP in first phase and phase Meta-RaPS Greedy in second, Meta-RaPS SAP in first phase and Greedy in second phase) were evaluated using large data sets ranging from 50 to 2000 workers and 55 to 3000 tasks. As previously mentioned, the optimal solution is not available for these data sets as they are too large to be solved in a reasonable amount of time by commercial software. The MR SAP version attains solution values (i.e. total training costs) much lower than those of the other methods, averaging 4.69% lower than those of the purely greedy, 2.93% lower than the MR Greedy.

Computer run times for these solution methodologies are obviously a function of the problem size. For example run times for 10,000 iterations of MR Greedy for a problem of size 9 workers, 17 tasks,and11 skills was 16.14 seconds on a Dell Inspiron I6400 PC with 1.00 GB of RAM.

In this chapter, the models, applications, methods and heuristics algorithms pertinent to the Assignment problem are presented. The skills management problem addressed in this research originally introduced in DePuy *et al*. (2006), is a type of the generalized assignment problem. The skills management model is a NP-hard problem. The exact solution methods only worked for relatively small problem sizes. A greedy algorithm and meta-heuristic solution approach for medium sized problems method suggested by DePuy *et al*. (2006, 2009) are investigated in this chapter. However, these existing solution approaches do not obtain the desired solution quality for large sized problems, i.e. hundreds of workers and tasks. In the next chapter three solution algorithms are developed and a data clustering method is applied for these large-sized skills management problems.

CHAPTER 3

SOLUTION METHODOLOGIES FOR LARGE SCALE SKILLS MANAGEMENT PROBLEMS

3.1. Introduction

As said in a previous chapter, feasible solutions of math models for the skills management problem could not be found in a reasonable amount of time for large sized problems. Therefore, developing a good heuristic solution methodology is critical for the proposed models.

In this chapter, four solution methods are developed for the skills management problem. The primary objective of these solution methodologies is to find high quality solutions for large sized problems. First, a model simplified method is implemented in an attempt to develop a good upper bound for the skills management problems. As mentioned previously, the reasons of complexity of the model are dependent costs and updating the skill levels in each training stage. So if this complexity could be simplified somehow, the SAP algorithm could be used to solve the problem easily. The main idea of the proposed heuristics methodologies is based on forming groups of tasks such that there are the same numbers of task groups as workers at which point task groups can be easily assigned to workers using SAP.

3.2. Finding a Good Upper Bound

Changing the formulation and simplifying some of the difficult constraints (those that cause the complexity of the problem) can lead to find a good upper bound for objective in much less time. This upper bound helps to determine the performance of other heuristics. One of the complexities of the skills management problem formulated in 2.3.1 is the updating of skill levels after each assignment that are shown in the (2-6) and (2-7) constraints, repeated here, and the following decision variables related to training.

$$S_{ik}W_{ik} + \sum_{m>s_{ik}}^{5} mZ_{iks_{ik}m} \geq R_{jk}X_{ij} \qquad \forall i,j,k \in \{j\} \qquad (3\text{-}1)$$

$$W_{ik} + \sum_{m>S_{ik}}^{5} Z_{iks_{ik}m} = 1 \qquad \forall i,k \qquad (3\text{-}2)$$

$$Z_{ikS_{ik}m} = \begin{cases} 1, & worker\ i\ receives\ training\ on\ skill\ k\ to\ raise\ skill\ level\ from \\ & Sik\ to\ m \\ 0, & otherwise \end{cases}$$

$$W_{ik} = \begin{cases} 1, & worker\ i\ does\ not\ need\ further\ training\ in\ skill\ k \\ 0, & otherwise \end{cases}$$

By changing these constraints and combining the decision variables to the following, the complexity of the problem decreases. The simplified model determines the necessary increase in skill level for each task, not over all tasks.

$$F_{iks_{ik}R_{ij}j} =$$

$$\begin{cases} 1, & worker\ i\ receives\ training\ on\ skill\ k\ to\ raise\ skill\ level\ from \\ & Sik\ to\ R_{ij}\ for\ doing\ task\ j \\ 0, & otherwise \end{cases}$$

$$(R_{jk} - S_{ik}) * F_{iks_{ik}R_{ij}j} \geq (R_{jk} - S_{ik}) * X_{ij} \qquad \forall i,j,k \in \{j\} \qquad (3\text{-}3)$$

Here, when worker *i* is assigned task *j* then X_{ij} is equal to 1 and if the required skill level k for task j (R_{jk}) is more than the skill level of the assigned worker *i* (S_{ik}), then $F_{ikS_{ik}R_{ij}j}$ has to be equal to 1.

The simplified 0-1 ILP model will be:

Minimize Training Cost: Minimize $\sum_i \sum_k \sum_m C_{kS_{ik}m} F_{ikS_{ik}R_{ij}j}$ (3-4)

Subject to:

Determine Needed Training:

$\left(R_{jk} - S_{ik}\right) * F_{ikS_{ik}R_{ij}j} \geq \left(R_{jk} - S_{ik}\right) * X_{ij}$ $\forall i, j, k \in \{j\}$ (3-5)

All tasks assigned: $\sum_i X_{ij} = 1$ $\forall j$ (3-6)

All workers assigned at least one task: $\sum_j X_{ij} \geq 1$ $\forall i$ (3-7)

Worker Capacity: $\sum_j T_j X_{ij} + \sum_k \sum_m E_{kS_{ik}m} F_{ikS_{ik}R_{ij}j} \leq A_i$ $\forall i$ (3-8)

Binary Variables:

$X_{ij} \in \{0,1\}$, $F_{ikS_{ik}R_{ij}j} \in \{0,1\}$ $\forall i, j, k, m$ (3-9)

This formulation can help to find the upper bound. Although there are more variables in this formulation, the model is simplified by combining some variables to new ones and disregarding updating skill levels in each worker training. It obtains an upper bound for the skills management model.

3.3. Sort Minimum to Maximum Cost Tasks (SMIMX)

It is known that the assignment problem can be solved quickly when the number of tasks is the same as the number of workers. When the number of tasks exceeds the number of workers, as is often the case in reality, the complexity and, hence, the solution time of the assignment problem grows. In these cases, grouping tasks into the same

number of groups as workers and using SAP (see section 2.3.2.3) can lead to find a good solution quickly. Several methods of grouping tasks are investigated in this research. The first task grouping method developed is the Sort minimum to Maximum Cost Tasks (SMIMX).

In SMIMX, tasks are grouped by similar total training costs and similar required levels of skills. The following steps describe the proposed SMIMX procedure to combine tasks that require similar levels of skills:

Step1: Each worker's skill and skill level is totally ignored. Instead, the main focus is on the nature of tasks and the required skills level to accomplish each task. The needed parameters from the original problem for the primary stage are:

R_{jk} = required skill level for task j's skill k

C_{klm} = cost associated with raising a worker's skill level on skill k from level l to level m

T_j = length (# hrs) of task j

The model has N workers and L tasks and P skills (N<L).

Step2: Calculate the total training cost of each task for a hypothetical least skilled worker (i.e. all skill levels of worker are 1)

Total training cost for task j

$$ToC_j = \sum_{k=1}^{p} C_{k1R_{jk}} \qquad \forall j \in L \text{ ……..(3-10)}$$

Step3: Sort the total training cost (ToC_j) from minimum to maximum (Labeled tasks in this order).

Step4: Consider the first N tasks with the minimum total training costs. Define a proper capacity for the time length for new task groups: TS_y (y=1, 2, …, N). This time length is defined in this way:

$$TS_y = \left\lceil \frac{L}{N} \right\rceil * \text{Max}_j\, T_j \qquad \forall y \in N \qquad (3\text{-}11)$$

Step5: Define the new task groups and update the last ordered list of defined tasks in this way:

For the each remaining (L-N) task do respectively these steps (h=N+1, N+2, …, L)

If ($T_j+T_h<TS_y$) then do next steps:

Step5-1: Calculate new task group's skill level k:

$$R'_{(j,h)k} = \text{Max}\{R_{jk}, R_{hk}\} \qquad \forall k \in p \qquad (3\text{-}12)$$

Step5-2: Calculate the new total training cost of each new task groups:

Total training cost for new task group v is (∀v=1,…, N):

$$TC'_v = \sum_{k=1}^{p} C_{k1R'_{(j,h)k}} \qquad (3\text{-}13)$$

Step5-3: Calculate the difference between total training cost of the new task groups and the previous defined tasks:

$$dif_v = TC'_v - ToC_v \qquad \forall v = 1, \dots, N \qquad (3\text{-}14)$$

Step5-4: Choose the minimum of difference cost. The index of this minimum is assumed "d".

Choose d $\ni dif_d = Min_v\{dif_v\}$ (3-15)

Step5-5: Update the d^{th} task group in this way:

$T_d=T_d+T_h$ (3-16)

$R_{dk} = \text{Max}\ \{R_{dk}, R_{hk}\}$ $\forall k \in p$ (3-17)

Total training cost for new defined task group d: $ToC_d = \sum_{k=1}^{p} C_{k1R_{dk}}$ (3-18)

Go back to step5.

Now there are N task groups and N workers. The SAP algorithm detailed in section 2.3.2.3 can be used to find the optimal solution for these combined tasks. The pseudocode for the SMIMX algorithm is shown in Figure 6. The complexity of the SMIMX method with N workers and L tasks with p skills is O (pNL) when L is larger than N.

```
Calculate total training cost for each task over hypothetical least skilled worker
Sort total_task_cost matrix
Fix the first minimum N tasks
Define a proper capacity for the length of time for N new future tasks
Do until all other (L-N) tasks are added to first N minimum tasks
        If (the length of time for doing both tasks< capacity)
                Define a new task by select the maximum level skill of each skill pair
                from both tasks
                Calculate total training cost for new task over hypothetical least skilled
                worker
                Calculate the difference of new total_task_cost of new task and the
                previous
                Select the minimum difference
                Update the skill level based on maximum of them
                Update total_task_cost
        Else
                There is no proper task to add, Stop
        End If
Loop
USE SAP for N workers and N tasks
```

Figure 6. Pseudocode for the SMIMX Algorithm

The SMIMX algorithm is developed to decrease problem complexities and form the problem in a way to use SAP for problem solving. But the feasibility of the problem cannot be ensured in the new format and in some cases, SAP does not find a feasible solution to the newly formed problem when the original problem may have feasible solutions. In some cases, the SMIMX defines new task groups that none of the workers can do them because of the time limitation. But the original problem may still have a feasible solution. So developing another new method considering workers capacities for

defining the new task groups may increases the chances of finding a feasible solution to the original problem.

3.4. Fix the most difficult task and combine tasks in new groups (MaxCT)

Grouping similar tasks with considering workers time limitation is the general idea of MaxCT method. In this procedure, N task groups are formed where N is the number of workers. The groups will initiate with the maximum total training cost tasks. The remaining tasks will be added to these N groups by considering changes to training costs and finding at least one worker who can be assigned to that group considering the worker's time limitation. At the end, having the same number of workers and tasks, the problem at hand can be solved by SAP algorithm (see section 2.3.2.3) to optimally assign the newly defined task groups to workers. The needed parameters from the original problem for the primary stage are:

S_{ik} = worker i's skill level for skill k

R_{jk} = required skill level for task j's skill k

T_j = length (# hrs) of task j

A_i = capacity (# hrs) of worker i

C_{klm} = cost associated with raising a worker's skill level on skill k from level l to level m

E_{klm} = time required (# hrs) to raise a worker's skill level on skill k from level l to level m

The model has N workers and L tasks and P skills (N<L).

The following steps describe the MaxCT procedure to combine tasks under some assumptions:

Step1: Define a hypothetical difficult task to use in step2 for sorting workers. The skill levels of this task are defined in this way:

IR_k: the required level for *k*th skill= $\text{Max}_j \{R_{jk}\}$ $\forall k \in p$ (3-19)

Step2: Sort workers from the most skilled to the least in this way:

Total training cost of worker i to be skilled:

$$WC_i = \sum_{k=1}^{p} C_{kS_{ik}IR_k} \qquad \forall i \in N \qquad (3\text{-}20)$$

Here, the worker with the lowest total training cost will be the most skilled (Labeled workers in this order).

Step3: Calculate the total training cost of each task for a hypothetical least skilled worker (i.e. all skill levels of worker are 1).

Total training cost for task j:

$$TtC_j = \sum_{k=1}^{p} C_{k1R_{jk}} \qquad \forall j \in L \qquad (3\text{-}21)$$

Sort the total training cost from maximum to minimum (Labeled tasks in this order).

Step4: Form N task groups. First put task number 1 (the most difficult) in the first task group.

Step5: Consider the top 50% percent of most skilled workers. Define the new task groups and update the previous task groups in this way:

For each remaining ungrouped task do these steps respectively:

Step5-1: Combine tasks with the previous task groups that are defined till now in this way and calculate their total training cost:

New task group's skill level k: $R'_{(j,f)k} = \text{Max}\{R_{jk}, R_{fk}\}$ (3-22)

Total training cost of task group: $TtC'_f = \sum_{k=1}^{p} C_{k1R'_{(j,f)k}}$ (3-23)

$\forall k \in p, \forall f \in N$ (Groups that are defined till now), j=f+1,…, L

Step5-2: Calculate the time of new task groups by summation of all time tasks in that group.

Step5-3: Check the top 50% of workers, consider their time limitation. If at least one worker can be found to be assigned to the new task groups, combine the tasks in the following sub steps otherwise go to step 6.

Step5-4: Calculate the difference between total training cost of the new task groups (the task groups that could have been found to assign some workers to) and the previous task group:

$\text{Difre}_f = TtC'_f - TtC_f$ (3-24)

Step5-4: Choose the minimum of difference cost. The index of this minimum is assumed "a":

Choose a $\ni difre_a = Min_f\{difre_f\}$ (3-25)

Step5-5: Update the task groups in this way:

$T_a=T_j+T_a$ (3-26)

T_a will be the new task group's time.

$R_{ak} = \text{Max}\ \{R_{ak}, R_{fk}\}$ $\forall k \in p$ (3-27)

Total training cost for new defined task group a: $TtC_a = \sum_{k=1}^{p} C_{k1R_{ak}}$ (3-28)

Go back to step 5.

Step6: Put task in the next empty task groups and go back to step5. If there are no empty task groups, then randomly choose 2 task groups and swap tasks until can find workers to assign. If this cannot be done, the procedure is failed.

Step7: After each $(\left\lceil\frac{L}{N}\right\rceil + \left\lceil\frac{2(L-N)}{N}\right\rceil)$ iteration, remove the top worker from the checking list and add another worker instead to the list. This process increases the accuracy of the whole procedure by not assigning workers to more than one task group.

Now when the MaxCT procedure results in N task groups then there are N tasks and N workers. The SAP algorithm can be used to find the optimal solution for these combined tasks. The pseudocode for the MaxCT algorithm is shown in Figure 7.

Sort workers from smallest to largest **total_training_cost** doing tasks

Consider the top 50% of most skilled workers in check list

Calculate total training cost for each task over a hypothetical least skilled worker

Sort from hardest to easiest task

Fix N positions for new tasks and put the hardest task in the first position

Do Until all tasks fill positions

Define a new task by select the maximum level skill of each pair

If a worker from the check list can be found to assigned

Select minimum change in cost

Update the task time

Update the skill levels

Else

If there is another empty position fill it

Else

If can find a worker to assign

Choose randomly 2 positions and swap tasks

End If

Else Stop

End If

End If

After each $(\left\lceil \frac{L}{N} \right\rceil + \left\lceil \frac{2(L-N)}{N} \right\rceil)$ iteration, remove the top worker from the checking list and add another worker instead to the list

Loop

USE SAP for N workers and N tasks

Figure 7 Pseudocode for the MaxCT Algorithm

The MaxCT algorithm is also developed to decrease problem complexities and form the problem in a way to use SAP. The ability of the procedure to find feasible solutions was increased with checking list of workers in sub step 5-3. The MaxCT is not guaranteed to find a feasible solution assuming one exists.

The method accuracy will increase in following cases:

1. Tasks are more similar to one another,
2. Number of skill levels are big,
3. Number of tasks are much more than number of workers,
4. Workers are at the same level of skills and not various in skills.

The problem is difficult to solve with this method when has the following characteristics:

1. Tasks are very different in their skills levels,
2. Skills and their levels are very different,
3. Workers are varied in their
4. Number of tasks and number of workers are close.

The complexity of the MaxCT method with N workers and L tasks with p skills is O(pNL).

3.5. Using *k*-means clustering method to classify tasks

Clustering is another method to group tasks into N groups (where N is number of workers) so that SAP can be used to do assignment of task groups to workers.

Clustering is a method of grouping data in such a way that objects will be similar (or related) to one another and dissimilar (or unrelated) to objects in other groups. It can

be formulated as a multi objective optimization problem. The proper clustering algorithm and parameter settings depend on the data set in hand and intended use of the results. There are many clustering algorithms that the most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another.

The "*k*-means" clustering algorithm is described in detail by Hartigan (1975). The *k*-means clustering method is a cluster analysis which partitions m object with p feature into *k* clusters. This *k* is specified by the user. The algorithm then randomly chooses *k* points from objects; these objects are the initial centers of the clusters. Each object is assigned to the center it is closest to. The *k*-means method is numerical, non-deterministic, and iterative. The problem is NP-Hard; but there are heuristic algorithms that can solve the problems quickly to a local optimum. It always has *k* clusters and each cluster has at least one object. The clusters are non-hierarchical and they do not overlap.

Tasks in the proposed assignment problem can be defined as a dataset with L objects. Each object has p features (here skills). The dataset is clustered into N clusters equal to the number of workers, and then SAP can be used to solve the assignment problem of N clusters of tasks to N workers. Although the clusters that are formed may not lead to a feasible solution, it may be helpful in large datasets. In this section, *k*-means clustering in MATLAB is used for the problem to cluster tasks with similar skills.

The data set with L tasks and p skills is given to the MATLAB to group tasks into N clusters (i.e. number of workers). L tasks with p skills are partitioned into N clusters by

heuristics method defined in MATLAB. The syntax used in MATLAB for k-mean is as follows:

[IDX, C, sumd, D]= kmean (X, N)

It partitions the points in the L-by-p data matrix X (L tasks with p skills) into N groups. It returns distances from each point (task) to every centroid in the L-by-N matrix D. Then the minimum element of each row (row's number indicates the index of task) in D is selected to group that task in the proper group. At the end, the problem is converted to one to one assignment problem and can be solved with the SAP algorithm. The complexity of this method with N workers and L tasks with p skills is the same as the previous ones, O (pNL). It should be noted, none of the proposed methods is not guaranteed to find a feasible solution assuming one exists.

CHAPTER 4

COMPUTATIONAL RESULT

4.1. Introduction

In chapter 3, four solution methods were developed. The upper bound, SMIMX, MaxCT, and *k*-means clustering discussed previously. The new methods are tested for randomly generated data sets (DePuy *et al.* 2007) and the results are compared with the previous results obtained from discussed methodologies in Chapter 2. Due to the problems' size, the optimal solution cannot be found for these data sets.

4.2. Results

The results from three previous solution methodologies, Greedy Assignment algorithm, Meta-RaPS Greedy Assignment and Meta-RaPS Shortest Augmenting Path (MR SAP) are included in tables 1 and 2 and 3 as well as the results from the proposed solution methods of upper bound (for small data sets), SMIMX, MaxCT, and *k*-means clustering.

In table 1 and 2, to investigate the performance of the solution methodologies for various ratios of workers and tasks, 26 small data sets were generated by DePuy *et al.* (2007). These data sets have 9 workers and 9 to 36 tasks with 11 skills and 5 skill levels and the other ones are randomly generated for 11 workers and 11 to 33 tasks with 13

skills and 5 skill levels. The seven solution methodologies were evaluated using these small data sets. The results are not very different for the new proposed methods in these small data sets. The differences in the solution values are not large and the average improvement is 0.3% for the new methods developed in this research. Figure 8 shows that in these small data sets, MaxCT outperforms the others. The best solution value of the proposed methods compared to upper bound results shows a good performance in the solution. On average, for the data set with 9 workers, the results are 16.02% better than the upper bound value and in the data set with 11 workers, this number is 18.49%.

TABLE 1

Results for Data sets with varied ratio of tasks to 9 workers

Data Set (Problem size)			1	2	3	4	5	6	7			
Number of workers	Number of tasks	Number of skills	Phase 1: MR SAP Phase 2: Greedy	Phase 1: Greedy Phase 2: Greedy	Phase 1: MR SAP Phase 2: MR Greedy	Upper Bound	SMIMX	MaxCT	k-mean clustering	Best Solution Value and Methods		Difference of the best value from Upper Bound(%)
9	9	11	391	391	391	391	391	391	391	391	all	0.00%
9	10	11	483	483	483	519	483	483	483	483	all except 4	6.94%
9	11	11	518	545	518	544	511	516	516	511	5	6.07%
9	12	11	527	547	527	609	545	540	527	527	1,3,7	13.46%
9	13	11	566	601	566	608	573	581	590	566	1,2	6.91%
9	14	11	593	642	593	717	601	590	610	590	6	17.71%
9	15	11	623	660	623	767	615	609	620	609	6	20.60%
9	16	11	628	653	624	779	627	621	640	621	6	20.28%
9	17	11	660	693	658	867	643	656	663	643	5	25.84%
9	18	11	697	734	690	950	703	710	709	697	1	26.63%
9	21	11	724	845	719	978	710	719	718	710	5	27.40%
9	36	11	905	1116	887	1118	896	890	910	890	6	20.39%
												16.02%

TABLE 2

Results for Data sets with varied ratio of tasks to 11 workers

Data Set (Problem size)			1	2	3	4	5	6	7			
Number of workers	Number of tasks	Number of skills	Phase 1: MR SAP Phase 2: Greedy	Phase 1: Greedy Phase 2: Greedy	Phase 1: MR SAP Phase 2: MR Greedy	Upper Bound	SMIMX	MaxCT	k-mean clustering	Best Solution Value and Methods		Difference of the best value from Upper Bound(%)
11	11	13	621	621	621	621	621	621	621	621	all	0.00%
11	12	13	611	611	611	663	611	611	611	611	all except 4	7.84%
11	13	13	616	624	616	716	613	616	616	613	5	14.39%
11	14	13	686	714	686	800	691	686	728	686	1,3,6	14.25%
11	15	13	675	705	675	891	689	694	681	675	1,3	24.24%
11	16	13	748	777	723	919	740	729	738	723	3	21.33%
11	17	13	756	796	753	964	767	761	775	753	3	21.89%
11	18	13	793	850	772	1018	798	769	790	769	6	24.46%
11	19	13	840	874	834	1125	851	839	846	834	3	25.87%
11	20	13	864	889	862	1201	857	852	869	852	6	29.06%
11	21	13	881	912	874	1227	856	860	879	856	5	30.24%
11	22	13	917	1018	908	1282	896	904	915	896	5	30.11%
11	28	13	1090	1325	1084	1134	1066	1053	1102	1053	6	7.14%
11	33	13	1286	1629	1271	1383	1290	1284	1324	1271	3	8.10%
												18.49%

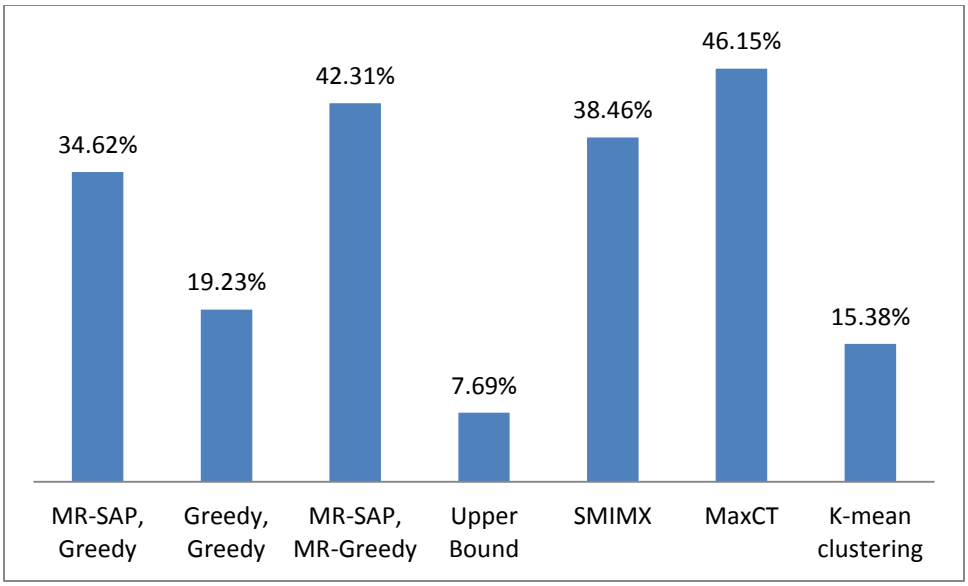

Figure 8. Percentage of the best solutions obtained by each algorithm in 26 small data sets

The proposed methods have been coded in MATLAB 7.12.0(R2012a). Computer run times for these solution methodologies are obviously a function of the problem size. As reported in previous literature, Jackson *et al* (2008), run times for 10,000 iterations of MR Greedy and MR SAP for a problem of size 9 workers, 17 tasks, and 11 skills was 16.14 seconds and 16.43 seconds respectively on a Dell Inspiron I6400 PC with 1.00 GB of RAM. Run times for suggested methods, SMIMX and MaxCT, for the same problem was 0.32 and 0.46 seconds respectively on a Dell Precision T1600 PC with 4.00 GB of RAM.

Table 3 shows results for 12 medium and large data sets ranging from 50 to 2000 workers and 75 to 3000 tasks. As previously mentioned, the optimal solution is not available for these data sets as they are too large to be solved in a reasonable amount of time by a commercial solution. Table 3 shows that even with large sized problems, the

MaxCT method still maintains its dominance over the other methods. This can be seen in the results for each tested large data set. The MaxCT attains solution values lower than those of the other methods. When comparing previous methodology to the MaxCT and SMIMX methods, it can be seen that they attain values averaging 7.34% lower than other previous methods. This is a good improvement and allows for great potential savings. As it can be seen the MaxCT outperforms in 67% and SMIMX in 33% cases. So overall, MaxCT outperforms the others in all data sets. Run times for large data sets on a Dell Precision T1600 PC with 4.00 GB of RAM using the proposed methods are shown in table 4. For example for a problem with 2000 workers and 2200 tasks and 50 skills, run times for SMIMX, MaxCT and k-mean clustering are 744, 878 and 323 seconds respectively. As shown in table 4, run times for MaxCT is more than other methods and K-mean clustering is faster. Figure 9 shows the run time comparison between these methods.

TABLE 3 Result for large data sets

Data Set (Problem size)			1	2	3	5	6	7			
Number of workers	Number of tasks	Number of skills	Phase 1: MR SAP Phase 2: Greedy	Phase 1: Greedy Phase 2: Greedy	Phase 1: MR SAP Phase 2: MR Greedy	SMIMX	MaxCT	k-mean clustering	Best Solution Value and Methods		Percent of Average Improvement
50	75	50	16089	16435	15394	15001	15109	16791	15001	5	2.55%
50	100	50	20600	20659	19436	17462	17305	21001	17305	6	10.96%
100	150	50	30089	30834	29452	26659	28891	30102	26659	5	9.48%
100	200	50	35413	36138	34501	29675	29431	35981	29431	6	14.70%
200	300	50	59045	60810	58055	57581	56003	N/A	56003	6	3.53%
200	400	50	71254	71835	69362	65041	67834	73908	65041	5	6.23%
500	550	50	134598	140688	134018	132079	128198	145892	128198	6	4.34%
500	750	50	144220	148238	141690	137491	141098	146701	137491	5	2.96%
1000	1100	50	264631	274574	263256	262405	258309	282109	258309	6	1.88%
1000	1500	50	281221	288939	279387	269724	268921	290184	268921	6	3.75%
2000	2200	50	521070	541494	519659	514286	503567	545781	503567	6	3.10%
2000	3000	50	549748	565549	547275	546421	535631	N/A	535631	6	2.13%
											7.34%

Table 4 Run time for Large Data Sets

Data Set (Problem size)			Run time		
Number of workers	Number of tasks	Number of skills	SMIMX	MaxCT	K-mean Clustering
50	75	50	10sec	12sec	3sec
50	100	50	12sec	15sec	4sec
100	150	50	25sec	30sec	10sec
100	200	50	28sec	34sec	10sec
200	300	50	47sec	54sec	32sec
200	400	50	50secs	60sec	32sec
500	550	50	140sec	200sec	63sec
500	750	50	185sec	272sec	114sec
1000	1100	50	410sec	462sec	192sec
1000	1500	50	463sec	512sec	237sec
2000	2200	50	744sec	878sec	323sec
2000	3000	50	870sec	950sec	397sec

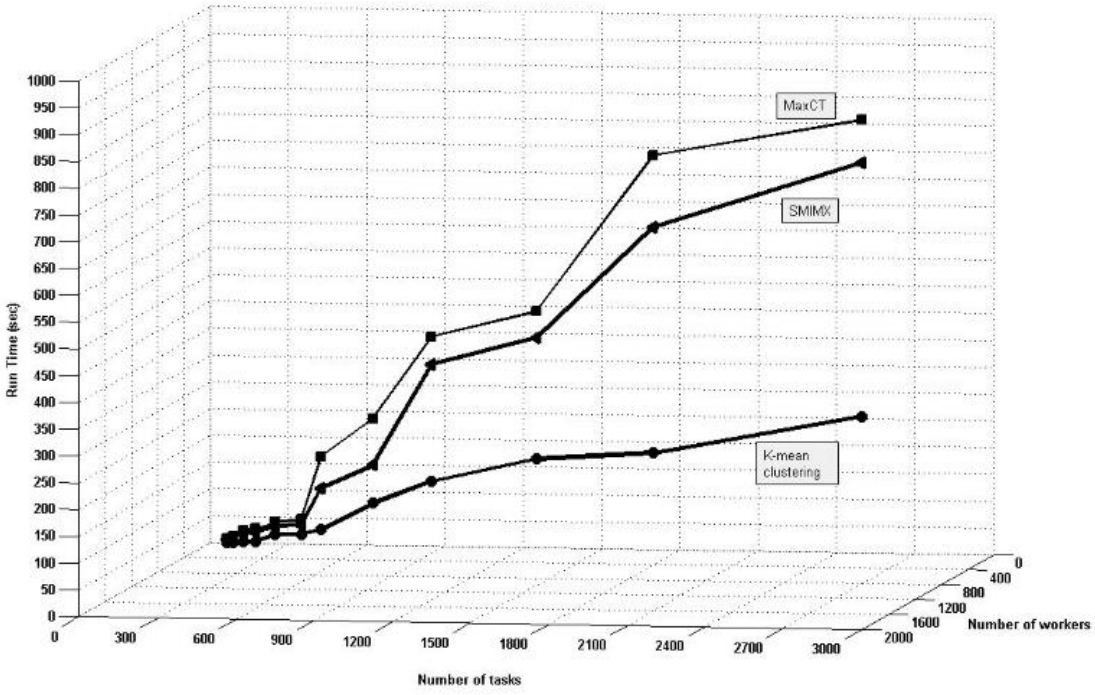

Figure 9 Run time comparisons for large data sets in proposed methods

CHAPTER 5

MULTI-OBJECTIVE SKILLS MANAGEMENT MODEL

5.1. Introduction:

The assignment of workers to tasks can lead to inefficient and ineffective job performance for several reasons. In the proposed skills management model, workers are assigned to tasks according their skills and trained to ensure effective and efficient job performance. But assigning workers to tasks against their preferences regarding the general condition of tasks may easily lead to reluctance. Consideration of skills and training workers and their preferences leads to higher motivation since workers are normally more interested to complete tasks related to their preferred abilities. The problem can be formulated as a multiple-objective problem, at once minimizing training costs and maximizing aggregate training subject to the previous constraints, and at the same time trying to maximize workers' preference to do tasks. In this chapter, multiple-objectives will be presented for the skills management problem to maximize workers preferences and aggregate training while minimizing training cost.

5.2. Assignment of workers to tasks under consideration of workers preferences

In companies throughout the world, an efficient assignment is an assignment that leads to the maximization of the preferences of workers. The presented model in this section seeks to develop assignments for workers to tasks according to their preferences and also aggregate training. Adding to previous parameters for the skills management model presented in 2.3.1, the preference of doing a task for each worker is defined by the workers themselves. The parameters for this model will be:

Prf_{ij} = worker i's preference to assign at task j

Other parameters are similar to the previous model, they are presented here again:

S_{ik} = worker i's skill level for skill k

R_{jk} = required skill level for task j's skill k

T_j = length (# hrs) of task j

A_i = capacity (# hrs) of worker i

C_{klm} = cost associated with raising a worker's skill level on skill k from level l to level m

E_{klm} = time required (# hrs) to raise a worker's skill level on skill k from level l to level m

The decision variables are defined as similar to the previous model:

$$X_{ij} = \begin{cases} 1, & worker\ i\ is\ assigned\ to\ task\ j \\ 0, & otherwise \end{cases}$$

$$Z_{ikS_{ik}m} = \begin{cases} 1, & worker\ i\ receives\ training\ on\ skill\ k\ to\ raise\ skill\ level\ from\ Sik\ to\ m \\ 0, & otherwise \end{cases}$$

$$W_{ik} = \begin{cases} 1, & worker\ i\ does\ not\ need\ further\ training\ in\ skill\ k \\ 0, & otherwise \end{cases}$$

The multi-objective model being proposed is:

Minimize Training Cost: Minimize $\sum_i \sum_k \sum_m C_{kS_{ik}m} Z_{ikS_{ik}m}$ (5-1)

Maximize Aggregate Training: Maximize $\sum_i \sum_k \sum_{m>s_{ik}} (m - S_{ik})\ Z_{ikS_{ik}m}$ (5-2)

Maximize Workers preferences: Maximize $\sum_i \sum_j prf_{ij} X_{ij}$ (5-3)

Subject to:

Determine Needed Training:

$$S_{ik}W_{ik} + \sum_{m>s_{ik}}^{5} mZ_{ikS_{ik}m} \geq R_{jk}X_{ij} \qquad \forall i, j, k \in \{j\} \qquad (5\text{-}4)$$

$$W_{ik} + \sum_{m>S_{ik}}^{5} Z_{ikS_{ik}m} = 1 \qquad \forall i, k \qquad (5\text{-}5)$$

All tasks assigned*:*

$$\sum_i X_{ij} = 1 \qquad \forall j \qquad (5\text{-}6)$$

All workers assigned at least one task:

$$\sum_j X_{ij} \geq 1 \qquad \forall i \qquad (5\text{-}7)$$

Worker Capacity:

$$\sum_j T_j X_{ij} + \sum_k \sum_m E_{kS_{ik}m} Z_{ikS_{ik}m} \leq A_i \qquad \forall i \qquad (5\text{-}8)$$

Binary Variables:

$$X_{ij} \in \{0,1\},\ Z_{ikS_{ik}m} \in \{0,1\},\ W_{ik} \in \{0,1\} \qquad \forall i, j, k, m \qquad (5\text{-}9)$$

The problem is solved for each of the three objectives (equations 5-1, 5-2, 5-3) and the decision maker is presented with all three decisions from which to choose.

Objective (5-2) maximizes the aggregate training for raising skill levels for workers and objective (5-3) maximizes the overall worker priority level

5.3. Multi-Objective Optimization Solution Methodologies

There are two general techniques to solve multi objective problems (Deb 2001). One approach is to combine all objectives into a single function or move all except one objective to the constraint set. The other approach is to generate a subset of efficient solutions or determine an entire Pareto optimal solution set (a set of solutions that are non-dominated by each other). The second method allows the decision maker to choose and evaluate the solutions.

There are many ways to transform a multi-objective problem to a single objective problem, including goal programming, weighted-sum approach, and ε-constraints programming.

Goal Programming was presented by Charnes et al. (1955). It attempts to find specific goal values of the objectives. The most commonly-used classical method is the ε-constraints method proposed by Chankong and Haimes (1983). It reformulates the multi-objective optimization problem to a single objective function.

As the name suggests, the weighted-sum approach changes a multi-objective optimization problem to a single-objective optimization problem by pre-multiplying each objective with a decision maker-supplied weight (Deb, 2001). This method is the simplest approach. It is probably the most widely used classical method.

In more detail, this method minimizes a positively weighted convex sum of the objectives and the solution depends on the importance of each objective in the context of the problem and a scaling factor. The scaling effect can be avoided somewhat by

normalizing the objective functions. After the objectives are normalized, a composite objective function can be formed by summing the weighted normalized objectives and the problem is then converted to a single-objective optimization problem as follows:

$$\text{minimize} \sum_{j=1}^{n} w_j f_j(x) \quad (5\text{-}10)$$

The second general solution approach for multi-objective optimization is to determine a representative subset solution, rather than get a single solution from every single run using the first general methods. The most widely-used heuristics methods for multi-objective optimization problem are genetic algorithms, simulated annealing, and tabu search. The algorithms of the above heuristics methods were initially developed for the single objective optimization problem, and are fit to solve multi-objective optimization.

5.4. Result for small data set

This multi-objective model is tested for some randomly generated data sets for small sized problems of 6 workers and 11 skills and 5 skill levels and 9, 10, 11, 12, 13 tasks. Also workers preferences are generated randomly. The results are compared for different hypothetical decision makers with different desire for objectives.

Weighted-sum method is used for solving the problem because this method is a simple approach. The weights are determined by the decision makers; in this research the weights are generated randomly. To normalize the objectives in these data sets, a scaling factor based on the optimal solution or upper bound (see section 3.2) of the problem for that single objective. So the final weighted single objective of the converted multi-objective problem for these data sets is as follows:

$$minimize\ \frac{w_1}{F_1^*}\left(\sum_i \sum_k \sum_m C_{kS_{ik}m} Z_{ikS_{ik}m}\right) - \frac{w_2}{F_2^*}\left(\sum_i \sum_k \sum_{m>s_{ik}} (m - S_{ik})\, Z_{ikS_{ik}m}\right) - \frac{w_3}{F_3^*}\left(\sum_i \sum_j p_{ij} X_{ij}\right) \quad (5\text{-}11)$$

Where F_i^* is the optimal value, if obtainable, or the upper bound as formulated in section 3.2 for the i^{th} objective individually.

These samples from the decision maker point of view are run in LINGO 11.0 on Dell Precision T1600 PC with 4 GB memory. The mathematical model for these samples was solved to optimality in seconds. The results are shown in table 4. For example, considering assignment of 6 workers to 11 tasks with 11 skills, if the decision maker's weight preferences for minimizing training cost, maximizing aggregate training, and maximizing workers preferences are 8, 1and 1, respectively, then the objectives will be 450, 72, and 32 respectively.

Table 4 Results for different decision makers with different values for small data sets with 6 workers

Data Set (Problem size)			Minimize Training Cost(w1,ob1), Maximize Aggregate Training(w2,ob2), Maximize Workers preferences(w3,ob3)								
Number of workers	Number of tasks	Number of skills	w1=1 w2=1 w3=1	w1=2 w2=2 w3=1	w1=3 w2=2 w3=1	w1=8 w2=1 w3=1	w1=1 w2=3 w3=2	w1=1 w2=8 w3=1	w1=1 w2=2 w3=3	w1=1 w2=2 w3=2	w1=1 w2=1 w3=8
6	9	11	ob1=308 ob2=60 ob3=32	ob1=306 ob2=60 ob3=29	ob1=303 ob2=57 ob3=29	ob1=303 ob2=57 ob3=29	ob1=341 ob2=75 ob3=32	ob1=663 ob2=129 ob3=34	ob1=314 ob2=66 ob3=32	ob1=314 ob2=66 ob3=32	ob1=320 ob2=64 ob3=34
6	10	11	ob1=301 ob2=63 ob3=36	ob1=299 ob2=61 ob3=36	ob1=295 ob2=57 ob3=36	ob1=398 ob2=70 ob3=35	ob1=332 ob2=76 ob3=36	ob1=648 ob2=128 ob3=37	ob1=305 ob2=67 ob3=36	ob1=305 ob2=67 ob3=36	ob1=318 ob2=62 ob3=40
6	11	11	ob1=321 ob2=65 ob3=40	ob1=317 ob2=65 ob3=35	ob1=311 ob2=59 ob3=35	ob1=450 ob2=72 ob3=32	ob1=347 ob2=77 ob3=40	ob1=648 ob2=128 ob3=38	ob1=326 ob2=70 ob3=40	ob1=326 ob2=70 ob3=40	ob1=374 ob2=74 ob3=48
6	12	11	ob1=319 ob2=63 ob3=44	ob1=320 ob2=68 ob3=39	ob1=311 ob2=59 ob3=39	ob1=311 ob2=59 ob3=39	ob1=374 ob2=86 ob3=44	ob1=627 ob2=125 ob3=44	ob1=326 ob2=70 ob3=44	ob1=326 ob2=70 ob3=44	ob1=379 ob2=73 ob3=51
6	13	11	ob1=365 ob2=67 ob3=46	ob1=366 ob2=72 ob3=41	ob1=359 ob2=65 ob3=41	ob1=359 ob2=65 ob3=41	ob1=428 ob2=92 ob3=49	ob1=657 ob2=127 ob3=49	ob1=389 ob2=79 ob3=49	ob1=376 ob2=75 ob3=46	ob1=418 ob2=76 ob3=55

CHAPTER 6

CONCLUSION AND FUTURE RESEARCH

6.1. Conclusions

In this dissertation, four solution methodologies are presented for the skills management problem of assigning tasks to workers, solved for several randomly generated data sets and the results compared with previous solution methods.

In the first method, upper bound, the complexity of the problem decreases by changing and simplifying some of the constraints and combining the decision variables. The simplified model determines the necessary increase in skill level for each task, not over all tasks. A good upper bound can be used to measure the performance of a heuristic method.

SMIMX and MaxCT are two other methods that decrease the complexity of the skills management problem by grouping tasks into N groups that can then be easily assigned to N workers. These proposed methods combine tasks that require similar levels of skills such that there is the same number of workers as there are task groups and then the SAP algorithm is used. Also *k*-means clustering is used for clustering tasks with similar skills in N cluster equal to the number of workers.

Several large-sized datasets are used to evaluate the performance of the heuristics. Results are compared to those of previous solution techniques as well as to the upper bound developed in this work. The new presented methods attain values averaging 7.34% lower than other previous methods in less than 16 minutes for large size problems.

Additionally the assignment problem with dependent costs is formulated as a multiple-objective optimization problem which maximizes workers preference at the same time optimizing two conflicting objectives, minimizing training cost while maximizing aggregate training. Several randomly generated data sets are tested for the proposed model and the results are compared for different hypothetical decision makers with different desire for objectives.

6.2. Future research

The assignment problem with dependent costs is classified as NP-hard problem; therefore there is no polynomial computational time method for solving problems. Future studies can include the following aspects relating to decreasing the run time while finding high quality solutions:

- ✓ Applying other heuristics and meta-heuristics to find a better solution from both time and objective quality points of view such as Genetic algorithms and Simulated annealing
- ✓ Improving the proposed methods, SMIMX and MaxCT, by considering workers skills levels and their time limitations

- ✓ Solving the revised multi-objective model for large data set and finding an efficient solution for the decision maker by heuristics and meta heuristics algorithm
- ✓ Using fuzzy membership functions for defining skill levels to have a more realistic model, and helping to solve it
- ✓ Try different and more proper distance function in MATLAB to form clusters.

REFERENCES

Amini, M., and Race, M., (1994). A rigorous computational comparison of alternative solution methods for the generalized assignment problem, *Management Science,* 40(7), 868-890.

Balachandran, V., (1972). An integer generalized transportation model for optimal job assignment in computer networks, Working Paper 34-72-3, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsbutgh.

Bozer, Y., and Carlo, H., (2007). Optimizing inbound and outbound door assignments in less-than-truckload cross-docks, *IIE Transactions,* 40, 1007-1018.

Campbell, G.M., Diaby, M., (2002), Development and evaluation of an assignment heuristic for allocating cross-trained workers, *European Journal of Operational Research* ,138, 9–20.

Cattrysse, D.G., Salomon, M., and Van Wassenhove, L.N., (1994). A set partitioning heuristic for the generalized assignment problem, *European Journal of Operational Research,* 72, 167-174.

Chankong V., yy Haimes: Multiobjective Decision Making Theory and Methodology, Elsevier Science Publishing Co., New York, 1983.

Charnes A, Cooper W W. Management Models and Industrial Applications of Linear Programming, Volume 1. New York: John Wiley, 1961

Chu, P.C. and Beasley, J.B.,(1997). A genetic algorithm for the generalized assignment problem, *Computers and Operations Research*, 24, 17-26.

Corominas, A., Pastor, R., and Rodriguez, E., (2006). Rotational allocation of tasks to multifunctional workers in a service industry, Int. J. Production Economics, 103, 3-9.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms.* Chichester: Wiley.

DePuy, G.W., Usher, J.S., and Jackson, E. (2008). "Logistics Skills Management Heuristics," Final Report for 2007-2008 CELDi Membership Project.

DePuy, G.W., Whitehouse, G.E. (2001). A Simple and Effective Heuristic for the Resource Constrained Project Scheduling Problem, *International Journal of Production Research*, 39(14), 3275-3287.

DePuy, G.W., Whitehouse, G.E., and Moraga, R.J., (2002). Using The Meta-Raps Approach to Solve Combinatorial Problems, CD-ROM *Proceedings of the 2002 Industrial Engineering Research Conference*, May 19-21, Orlando, Florida, 6 pages.

DePuy, G.W., Moraga, R.J., and Whitehouse, G.E. (2005). Meta-RaPS: A Simple And Effective Approach For Solving The Traveling Salesman Problem, *Transportation Research Part E: Logistics and Transportation Review*, 41(2), 115-130.

DePuy G.W., Usher, J.S., and Arterburn, B. (2006). "Workforce training schedule for logistics skills," CD-ROM *Proceedings of the 2006 Industrial Engineering Research Conference*, May 20-24, Orlando, Florida, 6 pages.

Dupont A., Linhares A., Artigues C., Feillet D., Michelon P., and Vasquez M. (2008). "The Dynamic Frequency Assignment Problem," *European Journal of Operational Research,* 195, 75-88.36

Douglas, A.M. (2006). A Modified Greedy Algorithm for the Task Assignment Problem. Master of Engineering thesis, University of Louisville.

Feltl. H, (2003) An Improved Hybrid Genetic Algorithm for the Generalized Assignment. PhD thesis, Vienna University of Technology, Vienna, Austria, 2003.

Fisher, M. and Jaikumar, R., (1981). A generalized assignment heuristic for vehicle routing, *Networks*, 11, 109-120.

Hasebrook, J., (2001). "Learning in the Learning Organization," *Journal of Universal Computer Science,* 1 (6), 472-487.

Hepdogan, S., R. Moraga, G.W. DePuy, and G.E. Whitehouse, (2009). "A Meta-RaPS Solution for the Early/Tardy Single Machine Scheduling Problem," *International Journal of Production Research,* 47(7), 1717-1732.

Jackson, E., DePuy, G.W., and Evans, G.E. (2007). "Logistics Skills Management Heuristics,". *Proceedings of the 2008 Industrial Engineering Research Conference*, May 17-21 Vancouver, Canada, 6 pages.

Jackson, E. (2009). SKILLS MANAGEMENT HEURISTICS. Master of Engineering thesis, University of Louisville.

Jonker, R. and Volgenant, A., (1987). "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," *Computing,* 38 (4), 325-340.

Kennington, J., and Wang, Z., (1992). "A Shortest Augmenting Path Algorithm for the Semi-Assignment Problem," *Operations Research,* 40 (1), 178-187.

Korossy, K., (1997). Extending the theory of knowledge spaces: A competence-performance approach. *Zeitschrift fur Psychologie,* 205, 53-82.37

Lan, G., DePuy, G.W. and Whitehouse G.E. (2007). An Effective and Simple Heuristic for the Set Covering Problem, *European Journal of Operational Research*, 176(3), 1387-1403.

Ley, T and Albert, D., (2003). "Identifying Employee Competencies in Dynamic Work Domains: Methodological Considerations and a Case Study," *Journal of Universal Computer Science,* 9 (12): 1500-1518.

Lorena, L.A.N, and Narciso, M.G., (1996). Relaxation heuristics for a generalized assignment problem, *European Journal of Operational Research,* 91, 600-610.

Martello, S., and Toth, P., (1981). An algorithm for the generalized assignment problem, *Proceedings of the 9th INFORS Conference,* Hamburg, Germany.

Mazzola, J., Neebe, A., and Dunn, C.,(1989). Production planning of a flexible manufacturing system in a material requirements planning environment, *International Journal of Flexible Manufacturing Systems*, 1, 115-124.

Moraga, R.J. (2002). Meta-RaPS An Effective Solution Approach for Combinatorial Problems, Ph.D. thesis. Orlando, FL: University of Central Florida.

Moraga, R.J., DePuy, G.W. and Whitehouse, G.E. (2005). Meta-RaPS Approach for the 0-1 Multidimensional Knapsack Problem, *Computers and Industrial Engineering*, 48(1), 83-96.

Moreno, N., Corominas, A., (2007). Solving the minsum product rate variation problem as an assignment problem, *Int. J. Flexible Manufacturing Systems,* 18, 269-284.38

Nasberg, M. and Jornsten, K., (1986). A new Lagrangian relaxation approach to the generalized assignment problem, *European Journal of Operational Research*, 27, 313-320.

Nauss, R.M., (2004). The Elastic Generalized Assignment Problem. *Journal of the Operations Research Society*, 55, 1333-1341.

Oncan, T. (2007). A Survey of the General Assignment Problem and It's Applications," *INFORMS*, 45 (3), 123-141.

Pentico, D. W. (2007), Assignment problems: A golden anniversary survey, *European Journal of Operational Research*, 176, 774–793

Rainwater, C., Geunes, J., & Romeijn, H. (2009). The General Assignment Problem with Flexible Jobs, *Discrete Applied Mathematics*, 157, 49-67.

Rabadi, G., Moraga, R., Al-Salem, A. (2006). Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times, *Journal of Intelligent Manufacturing*, 17(1), 85-97.

Ross, G.T., and Soland, R.M., (1975). A Branch and Bound Approach for the generalized assignment problem, *Mathematical Programming*, 8(1), 91-105.

Savelsbergh, M., (1997)., A branch-and-price algorithm for the generalized assignment problem, *Operations Research*, 45, 831-840.

Thorndike, R. L., (1950), The problem of the classification of personnel, *Psychometrika*, 15, 215-235.

Wilson, J.M., (1997). A simple dual algorithm for the generalized assignment problem, *Journal of Heuristics*, 2, 303-312.39

Yagiura, M., Ibaraki, T., and Glover, F., (2004). An ejection chain approach for the generalized assignment problem, *INFORMS Journal on Computing*, 2, 16-28.

APPENDIX

MATLAB code for SAP

```
function [rowsol,cost,v,u,costMat] = lapjv(costMat,resolution)

if nargin<2
    maxcost=min(1e16,max(max(costMat)));
    resolution=eps(maxcost);
end
 [rdim,cdim] = size(costMat);
M=min(min(costMat));
if rdim>cdim
    costMat = costMat';
    [rdim,cdim] = size(costMat);
    swapf=true;
else
    swapf=false;
end
dim=cdim;
```

```
costMat = [costMat;2*M+zeros(cdim-rdim,cdim)];
costMat(costMat~=costMat)=Inf;
maxcost=max(costMat(costMat<Inf))*dim+1;
if isempty(maxcost)
    maxcost = Inf;
end
costMat(costMat==Inf)=maxcost;
v = zeros(1,dim);
rowsol = zeros(1,dim)-1;  colsol = zeros(dim,1)-1;


if std(costMat(:)) < mean(costMat(:))
```

```
numfree=0;
free = zeros(dim,1);
matches = zeros(dim,1
for j=dim:-1:1

   [v(j), imin] = min(costMat(:,j));
   if ~matches(imin)
      rowsol(imin)=j;
      colsol(j)=imin;
   elseif v(j)<v(rowsol(imin))
      j1=rowsol(imin);
      rowsol(imin)=j;
      colsol(j)=imin;
      colsol(j1)=-1;
   else
      colsol(j)=-1;
   end
   matches(imin)=matches(imin)+1;
end

  for i=1:dim
   if ~matches(i)
      numfree=numfree+1;
```

```
            free(numfree)=i;
        else
            if matches(i) == 1
                j1 = rowsol(i);
                x = costMat(i,:)-v;
                x(j1) = maxcost;
                v(j1) = v(j1) - min(x);
            end
        end
    end
else
    numfree=dim-1;
    [v1 r]=min(costMat);
    free=1:dim;
    [~,c]=min(v1);
    imin=r(c);
    j=c;
    rowsol(imin)=j;
    colsol(j)=imin;
    free(imin)=[];
    x = costMat(imin,:)-v;
    x(j) = maxcost;
    v(j) = v(j) - min(x);
```

```
end
loopcnt = 0;
while loopcnt < 2
   loopcnt = loopcnt + 1;


   k = 0;
   prvnumfree = numfree;
   numfree = 0
   while k < prvnumfree
      k = k+1;
      i = free(k);
      x = costMat(i,:) - v;
      [umin, j1] = min(x);
      x(j1) = maxcost;
      [usubmin, j2] = min(x);
      i0 = colsol(j1);
      if usubmin - umin > resolution
        v(j1) = v(j1) - (usubmin - umin);
      else
         if i0 > 0
            j1 = j2;
            i0 = colsol(j2);
         end
```

```
        end
        rowsol(i) = j1;
        colsol(j1) = i;
        if i0 > 0
            if usubmin - umin > resolution
                free(k)=i0;
                k=k-1;
            else

                numfree = numfree + 1;
                free(numfree) = i0;
            end
        end
    end
end

for f=1:numfree
    freerow = free(f);
    d = costMat(freerow,:) - v;
    pred = freerow(1,ones(1,dim));
    collist = 1:dim;
    low = 1;
    up = 1;
```

```
unassignedfound = false;
while ~unassignedfound
   if up == low
      last = low-1;
      minh = d(collist(up));
      up = up + 1;
      for k=up:dim
         j = collist(k);
         h = d(j);
         if h<=minh
            if h<minh
               up = low;
               minh = h;
            end

            collist(k) = collist(up);
            collist(up) = j;
            up = up +1;
         end
      end

      for k=low:up-1
         if colsol(collist(k)) < 0
```

```
            endofpath = collist(k);
            unassignedfound = true;
            break
        end
    end
end
if ~unassignedfound
    j1 = collist(low);
    low=low+1;
    i = colsol(j1);
    x = costMat(i,:)-v;
    h = x(j1) - minh;
    xh = x-h;
    k=up:dim;
    j=collist(k);
    vf0 = xh<d;
    vf = vf0(j);
    vj = j(vf);
    vk = k(vf);
    pred(vj)=i;
    v2 = xh(vj);
    d(vj)=v2;
    vf = v2 == minh;
```

```
        j2 = vj(vf);
        k2 = vk(vf);
        cf = colsol(j2)<0;
        if any(cf)
            i2 = find(cf,1);
            endofpath = j2(i2);
            unassignedfound = true;
        else
            i2 = numel(cf)+1;
        end
        for k=1:i2-1
            collist(k2(k)) = collist(up);
            collist(up) = j2(k);
            up = up + 1;
        end
    end
end
j1=collist(1:last+1);
v(j1) = v(j1) + d(j1) - minh;
while 1
    i=pred(endofpath);
    colsol(endofpath)=i;
    j1=endofpath;
```

```
        endofpath=rowsol(i);
        rowsol(i)=j1;
        if (i==freerow)
            break
        end
    end
end
rowsol = rowsol(1:rdim);
u=diag(costMat(:,rowsol))-v(rowsol)';
u=u(1:rdim);
v=v(1:cdim);
cost = sum(u)+sum(v(rowsol));
costMat=costMat(1:rdim,1:cdim);
costMat = costMat - u(:,ones(1,cdim)) - v(ones(rdim,1),:);
if swapf
    costMat = costMat';
    t=u';
    u=v';
    v=t;
end
if cost>maxcost
    cost=Inf;
end
```

CURRICULUM VITAE

Ghazal Tariri

Work address: Department of Industrial Engineering
J.B. Speed School of Engineering
University of Louisville
Louisville, KY 40292

Education:

PhD, Industrial Engineering
University of Louisville
2009- Defended on July 25, 2013

Master of Science, Industrial Engineering
Isfahan University of Technology
2007-2009

Bachelor of Engineering, Industrial Engineering
Sharif University of Technology
2003-2007

Language Skills:

English (Fluent)

Persian (Native fluent)

Arabic (Fluent)