5-2009

Skills management heuristics.

Erin Jackson
University of Louisville

SKILLS MANAGEMENT HEURISTICS

By

Erin Jackson
B.S., University of Louisville, 2007

A Thesis
Submitted to the Faculty of the
University of Louisville
J.B. Speed School of Engineering
in Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Industrial Engineering

May 2009

LOGISTICS SKILLS MANAGEMENT HEURISTICS

Submitted by: ___________________________________
Erin Jackson

A Thesis Approved On

____
(Date)

by the Following Reading and Examination Committee:

____
Dr. Gail DePuy, Thesis Director

____
Dr John S. Usher

____
Dr. Eric C. Rouchka

ACKNOWLEDGEMENTS

I would like to thank Marla Fredrick, Barbara Strahley, and Randy Walker from the Crane division of the Naval Surface Warfare Center for their financial support of this research project as well as providing an interesting real world topic. Also thank you to the members of my thesis committee, Dr. John S. Usher and Dr. Eric C. Rouchka for taking the time to assist me in my endeavors.

Thank you to my family for being there to encourage and support me throughout my collegiate career. And last, but definitely not least, a special thanks goes to my thesis advisor Dr. G. W. DePuy for her invaluable guidance on this project as well as others.

ABSTRACT

A common problem faced by most organizations in today's world is one of worker-task assignments. Assigning a large number of complex tasks to workers at various training levels can be a complicated process which has the potential to cost or to save a company large sums of money. The aim of this project is to develop a heuristic tool designed to match tasks to workers given the workers' skills proficiency profiles. This heuristic should also provide a training plan which will rectify current worker skills gaps while minimizing training costs. Prior research maintained a focus on utilizing mathematical models of this skills management problem. The main difficulty with these mathematical models is that they were unable to reach feasible solutions in a reasonable amount of time when the problem size became large. It is therefore wise to investigate possible heuristic solution techniques. This research will compare and contrast three specific heuristic techniques: a Greedy Assignment Algorithm, Meta-RaPS Greedy Heuristic, and Meta-RaPS Shortest Augmenting Path (SAP) Heuristic. Meta-RaPS is a meta-heuristic that is used to improve the performance of algorithms by strategically infusing randomness which allows the exploration of more of the solution space.

The skills management heuristics developed in this research were tested using 47 randomly generated data sets generating results within 0.03% of optimal for the recommended Meta-RaPS SAP solution methodology.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

There are many problems that management teams face in every company throughout the world. One large issue is that of worker-task assignments. Assigning specific tasks to workers with varying skill levels is a complicated process that can have a large monetary impact on any company. It is important to approach this problem with great care and attention. The main cost that comes into play when solving the worker-task assignment problem is that of worker training. This is because if the workers are not trained in the necessary skills, they will be unable to complete the tasks necessary to the company's survival.

It is also important to realize that if worker-task assignments are not completed properly, there can be additional costs incurred. These costs include that of poor quality, and backlogged work. When poor worker-task assignments are made, it is likely workers will be assigned tasks for which they are not properly trained which will result in faulty work ultimately costing the company money. Also, it is a large possibility that improper assignments would result in qualified workers being overloaded with too many tasks and not enough capacity to complete them all which would result in a backlog of work, also costing the company money.

It is therefore important to consider methods that aid in the worker-task assignment model. In the past, these assignments are often completed manually by the management teams within the companies. This is a valid method of assignment as long as the team knows what criteria need to be met. The problem with manual evaluation of this issue is that as the number of workers and the number of tasks increase, the complexity of the problem grows rapidly and it becomes increasingly difficult to find high quality solutions. A solution technique which produces near optimal solutions for large-scale problems is needed.

The issue of turn-over also comes into play. Many companies not only have frequent worker turn-over, but also product turn-over as well. Each time new workers are introduced in the place of old workers, the process must change. This is also true when new tasks (or products) are introduced into the company of changes are made to the current tasks. Each change will cause added difficulty in manually configuring the worker-task assignments. An automated solution technique is recommended to allow solutions to be generated rapidly.

As previously stated, the main cost factor to be dealt with during the proper worker-task assignment is that of training. Since workers must be trained in order to complete these tasks, and training requires both time and money, a cost will be incurred for every worker that must complete some training. Therefore, these assignments should be made so that the overall amount of training is minimized in order to minimize the total cost of the worker-task assignment. Due to all of these issues, it has become apparent that a computer tool to aid in the worker-task assignment problem would be very useful as well as applicable to many companies throughout various industries.

The worker-task problem is discussed in greater detail in Chapter II. Chapter III is a review of the literature that is relevant to this application. The solution methodology will be described in Chapter IV followed by the results in Chapter V. Chapter VI, the final chapter, contains the conclusions obtained from this research as well as suggestions for future research.

II. PROBLEM DESCRIPTION

A detailed description of the worker-task assignment problem considered in this research is presented in this chapter. First terminology used throughout this paper is defined, then both a narrative and mathematical description of the problem is presented. There are three factors associated with work that are necessary to define to accurately describe this project. The first factor is referred to as *tasks* which are the specific jobs that will need to be completed. A generalized example of a task would be to change a flat tire. Secondly, *skills* must be considered. Skills are the specific abilities that a worker must have in order to complete a task. Continuing with the example of changing a flat tire, required skills 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. Lastly, the *level* of skills must be considered. Levels could be defined in simple terms such as novice, intermediate, and expert. Therefore, each *task* requires certain *levels* of various *skills* in order to be brought to fruition. For example, some cars may have more complex jacks or very tight lug nuts. These issues would require more advanced skills than others.

Furthermore, it is important to identify the assignment relationship between these three factors. Although skills are unique among themselves, various tasks may require the same skill or set of skills, at the same or different levels. For example, it may also be necessary to have the skill defined as “knowledge of undercarriage of car” when changing the oil on a car which would be considered a different task. Each task must be assigned to a worker. Although each task is only assigned to one worker, each worker may themselves be assigned multiple tasks.

Each task has its own set of skills, each with a specific skill level required. Likewise, each worker will have their own set of skills on which they have been trained to a specific level. The skill set of the worker is determined by the supervisor in charge of that worker. When a worker does not meet the skill level required by the task assigned to them, a *skills gap* is said to be present. In these instances the worker must be trained in order to correct the skills gap. As stated above, training requires both time and money to complete. A feasible solution to this skills management problem is one in which all workers have enough capacity (i.e. time) to complete all the tasks, and resulting training, assigned to them.

It is important to note that this research represents an assignment problem with dependent costs. This aspect makes this topic different from the general assignment problem. The main variation in this model is that it assumes the once a worker is trained to a certain level for a specific task, their level of skill is increased for all additional tasks that will be assigned to them. In other words, once a worker is trained to complete one task, that training will carry over which has the potential to cut down on any further training needed for additional tasks that may be assigned.

Additionally, the skills management problem addressed in this research includes the stipulation that all workers must be assigned at least one task. This 'workforce preservation' requirement ensures that no workers will be terminated. This skill management problem can be described mathematically as originally introduced in DePuy *et al.* (2006) and repeated here.

Parameters:

{j} = set of skills needed to perform task j

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

Decision Variables:

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

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

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

Objective Function:

Minimize Training Cost

$$\text{Minimize } \sum_i \sum_k \sum_m C_{kS_{ik}m} Z_{ikS_{ik}m} \qquad (1)$$

Constraints:

Determine Needed Training

$$S_{ik} N_{ik} + \sum_{m>S_{ik}}^{5} m Z_{ikS_{ik}m} \geq R_{jk} X_{ij} \qquad \forall i, j, k \in \{j\} \quad (2)$$

$$N_{ik} + \sum_{m>S_{ik}}^{5} Z_{ikS_{ik}m} = 1 \qquad \forall i, k \quad (3)$$

All tasks assigned

$$\sum_i X_{ij} = 1 \qquad \forall j \quad (4)$$

All workers assigned

at least one task

$$\sum_j X_{ij} \geq 1 \qquad \forall i \quad (5)$$

Worker Capacity $$\sum_j T_j X_{ij} + \sum_k \sum_m E_{kS_{ik}m} Z_{ikS_{ik}m} \le A_i \qquad \forall i \quad (6)$$

Binary Variables $$X_{ij} \in \{0,1\}, \quad Z_{ikS_{ik}m} \in \{0,1\}, \quad N_{ik} \in \{0,1\} \qquad \forall i, j, k, m \quad (7)$$

The first equation in the model is the objective function which dictates that the model be run in order to minimize the overall training cost of the assignment. Next, equations 2 and 3 are used to determine the total training needed by a worker in order to meet the skill levels required to complete a specific task. Equation 4 is used to ensure that all tasks have been assigned and that each task is assigned to only one worker. Equation 5 specifies that each worker must be assigned at least one task. The sixth equation makes sure the total workload assigned to a worker (i.e. task time plus training time) does not exceed the worker's capacity. Finally, equation 7 defines all decision variables to be binary.

As an extension to the original problem, this research will also include the capability to allow fixed assignments. There will be instances where the supervisor knows which employee needs to be assigned a specific task. In these cases, the supervisor can specify these "fixed assignments" before allowing the heuristics to make the remaining worker-task assignments. Mathematically, including fixed assignments would add the following constraints to the formulation presented above for the fixed assignment of worker i to task j.

$$X_{ij} = 1 \qquad (7)$$

There are three different solution heuristics developed in this research. First the Greedy Assignment Algorithm will be presented as a basis for the worker-task assignment problem. Next, a modified version of the Greedy Assignment Algorithm will be presented and is referred to as the Meta-RaPS Greedy heuristic as presented in DePuy *et al.*, 2006. Lastly, the Shortest Augmented Path (SAP) Algorithm will be modified via the meta-heuristic Meta-RaPS and applied to this skills management problem. An automated computer tool which implements each of the three heuristic solution methodologies is also developed.

III. LITERATURE REVIEW

This section reviews literature relevant to both the skills management problem and solution techniques presented in this thesis. The skills management problem formulated in the previous chapter is a variation of the generalized assignment problem which will be reviewed here as well.

<u>1. The Generalized Assignment Problem</u>

The Generalized Assignment Problem (GAP) is a combinatorial optimization problem that has been studied for many years. This type of problem occurs when it is necessary to assign required tasks to available resources. The GAP can be applied to many real world applications and is often described with two specific examples. The first is the knapsack problem which describes the GAP in terms of *items* of certain *weights* that must be placed into *knapsacks.* Also, the scheduling problem depicts the GAP as a problem where *jobs* with specific *processing times* are assigned to *agents* or *employees* (Oncan, 2007). Many have also expanded the general assignment problem to incorporate other factors such as profit maximization (Rainwater *et al.,* 2009), minimal training costs (DePuy *et al.*, 2008), and elastic costs (Nauss, 2004).

The solution methodologies for these types of problems are as varied as the problems themselves. These solution methodologies include the greedy assignment algorithm (Martello *et al.,* 1981), Branch and Bound (Ross *et al.,* 1975), Tabu (Dupont *et al.*, 2008), as well as many heuristic methods developed to handle specific versions of the general assignment problem. These heuristic methods include Set Partitioning Heuristic (Cattrysse *et al.,* 1994),Variable Depth Search Heuristic (Amini *et al.,* 1994) and Lagrangian Relaxation Heuristics (Lorena *et al.,* 1996).

The assignment problem with dependent costs is a special case of the GAP. In these cases the jobs which have already been assigned to workers are taken into account when assigning future tasks. This means that the workers capacity will be changed before any secondary assignment can occur and the training level of that worker will also be updated to correspond to the training received in order to complete the first task assigned. To date, similar problems have not appeared in the literature.

2. Skills Management

Skills management originates from the need to fulfill human resource constraints set forth by human resource departments with employees who compliment the company strategy (Ley, 2003). More specifically, skills management deals with formulating a list of competencies required by various jobs and assigning values to employees based on the level each has achieved for those specific competencies.

Many have worked to perfect the science of assigning competency levels to employees as well as finding the most useful training techniques. These include Skills Management Information Systems (SMIS) developed by J. Hasebrook (2001) as well as the Competence Performance Theory which was initially studied by Korossy (1997) and was further evolved by Ley and Albert (2003). Overall, one thing that everyone seems to agree upon is that a solid skills management method is crucial to any company who wishes to keep an up-to-date workforce.

These issues cover only the formulation for this problem. Once the problem has been defined and is understood the next step is to establish an adequate solution methodology. Much research has also been done in order to find and improve upon solution techniques for the GAP and DGAP. This research utilizes both a version of a

Metaheuristic for Randomized Priority Search (Meta-RaPS) and the Shortest Augmenting Path (SAP) algorithm.

3. Meta-RaPS

Meta-RaPS (Meta-heuristic for Randomized Priority Search) is a generic, high level strategy used to modify construction heuristics based on the insertion of randomness (DePuy and Whitehouse, 2001; DePuy *et al.*, 2002). Meta-RaPS integrates priority rules, randomness, and sampling. At each iteration, Meta-RaPS constructs and improves feasible solutions through the utilization of construction heuristic priority rules used in a randomized fashion. After a number of iterations, Meta-RaPS reports the best solution found. As with other meta-heuristics, the randomness represents a device to avoid getting stuck in a local optima.

Meta-RaPS has been applied to a variety of combinatorial problems such as: the Set Covering Problem (Lan *et al.*, 2007), the Unrelated Parallel Machine Problem (Rabadi *et al.*, 2006b), the Traveling Salesperson Problem (DePuy *et al.*, 2005), the Knapsack Problem (Moraga *et al.*, 2005), the Vehicle Routing Problem (Moraga, 2002), machine scheduling (Hepdogan *et al.*, 2009) and the Resource Constrained Project Scheduling Problem (DePuy and Whitehouse, 2001). Meta-RaPS 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 procedure will be discussed in greater detail in Chapter IV.

4. Shortest Augmenting Path (SAP)

The Shortest Augmenting Path algorithm has also been used to help solve versions of the general assignment problem. This algorithm has been proven to provide optimal results in a much faster time than other algorithms (Jonker and Volgenant, 1987) as well as faster times than many heuristics used in assignment software (Kennington and Wang, 1990). In their paper published in 1987, Jonker and Volgenant presented the use of the SAP algorithm for the linear assignment problem. Also, in that paper was included a Pascal code for the use of the SAP algorithm which was used as a basis for a portion of this project.

The SAP algorithm has since been used in a wide variety of applications. Examples or research that stems from the SAP algorithm includes: the allocation of tasks to multifunctional workers (Corominas *et al.,* 2006), trailer-to-door assignments with cross-docking (Bozer, 2007), and the solution of the minimum product rate variation problem (Moreno, 2007).

This research pertains to a problem based on the general assignment problem with dependent costs incorporating skill management information. Also in this research are solution methodologies which incorporate a version of the Meta-RaPS heuristic as well as the Shortest Augmenting Path algorithm. More in depth information on the specifications of the problem researched here can be found in the previous chapter (Chapter II) while additional information on the solution methodologies is presented in the following section (Chapter IV).

IV. SOLUTION METHODOLOGY

1. Greedy Assignment Algorithm

Typically a greedy algorithm builds a solution by iteratively adding feasible components to the solution until a stopping criteria is met. For this Greedy Assignment algorithm, tasks are iteratively assigned to workers until all the tasks have been assigned. Tasks are assigned such that the worker's capacity is not exceeded, each worker is assigned at least one task, and all the tasks have been assigned. The objective is to minimize the total training costs.

The Greedy Assignment algorithm developed for this skills management problem uses a two-phase approach. The first phase assigns exactly one task to each worker, and the second phase assigns any remaining tasks to workers with unfilled capacity. Phase 1 of the Greedy Assignment algorithm is used to maintain workforce preservation by ensuring each worker gets at least one task assigned to them. When considering fixed assignments, those fixed assignment are made before phase one 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.

The first phase of the greedy algorithm assigns the tasks to the workers beginning with the least skilled worker. This is determined by finding the total training cost for each worker to complete all the tasks. Those workers that are not skilled will require much training to complete all tasks and will therefore have a high total training cost. The workers are sorted from highest to lowest total training costs. Starting with the worker with the highest total training cost (i.e. the least skilled worker), workers are assigned the task which is easiest (i.e. lowest training cost) for them. Once a task is assigned, it is removed from consideration. At the end of phase 1 each worker is assigned exactly 1 task. The worker capacities and skills set are updated based on these phase 1 assignments. After phase one is complete, phase two assigns all remaining tasks to workers.

In phase 2, the remaining, unassigned tasks are ordered from the most difficult task to least difficult task as determined by the total training cost for all workers to complete the task. Those tasks with a high total training time are difficult tasks as many workers would require additional training to be able to complete the task. Starting with the task with the highest total training cost (i.e. the most difficult task), tasks are assigned to the worker which requires the least amount of training (i.e. lowest training cost) to complete the task. Once a task is assigned, it is removed from consideration and the workers 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). While the Greedy Assignment method guarantees a feasible solution, it has a tendency to become stuck at local optima and therefore can deviate 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 optima.

```
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. Meta-RaPS Heuristic

Meta-RaPS (Meta-heuristic for Randomized Priority Search) is a generic, high level strategy used to modify greedy algorithms based on the insertion of a random element (DePuy *et al.,* 2002). Meta-RaPS constructs feasible solutions through the utilization of a greedy algorithm in a randomized fashion. As with other meta-heuristics, the randomness represents a device to help avoid getting stuck in local optima. The general Meta-RaPS procedure will be reviewed below, then the specific application of Meta-RaPS to this skills management problem will be discussed.

The Meta-RaPS heuristic utilizes two parameters that are specified by the user in order to incorporate this randomness into the system: %priority and %restriction. The Meta-RaPS heuristic calculates all of the total training cost values for the workers and the tasks the same as the Greedy Assignment Algorithm did. The difference is in how it chooses to assign the tasks to the workers using these two user-defined parameters. The %priority parameter dictates how often the assignment specified by the Greedy Assignment Algorithm will be made versus when an assignment that is close to the greedy assignment will be made. Some of the time (i.e. 100%-%priority) and assignment whose cost is within %restriction of the cost of the greedy algorithm assignment will be made instead. 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.

The addition of these parameters and the randomness that they incur in the model allows the heuristic to avoid becoming stuck in local optima. 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 phase 1 and phase 2 of the Meta-RaPS Greedy Assignment heuristic.

```
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

```
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

In order to determine what values to set for the %priority and %restriction parameters, trials were completed. Both large and small data sets were tested using a wide range of parameter values. Each data set was run multiple times in order to determine which parameter settings aided the heuristic to the best solution value most consistently. The following parameters were determined to be most applicable to this problem: phase 1 %restriction of 500%, phase 2 %priority of 20%, and phase 2 % restriction of 20%.

3. Shortest Augmenting Path (SAP) Algorithm

Also investigated in this research is the inclusion of the Shortest Augmenting Path Algorithm (SAP) developed by Jonker and Volgenant (1987). SAP is an algorithm to find the optimal solution to the classic assignment problem (i.e. optimal assignment of n workers to n tasks).

For this research, it is assumed (as is in most companies) that there are more tasks than there are workers, i.e. workers will be assigned multiple tasks to complete. However, the SAP algorithm assigns an equal number of tasks to workers. To resolve this issue, the SAP algorithm is only incorporated into phase 1 of the algorithm (i.e. the assignment of 1 task to each worker). When the number of tasks exceeds the number of workers, a decision must be made as which subset of tasks will be assigned via SAP. The Meta-RaPS technique is used again to select from a list of 'easy' tasks (i.e. low total training cost tasks) those tasks which will be assigned in phase 1.

The SAP algorithm utilizes a series of four segments: (1) Initialization, (2) Termination, (if all rows are assigned), (3) Augmentation and (4) Adjustment of the dual solution. Also, the Initialization segment in itself contains a series of three 'sub-procedures'. These include: (1) column reduction, (2) reduction transfer (from unassigned rows to assigned rows), and (3) augmenting reduction of unassigned rows (Jonker and Volgenant, 1987).

The column reduction stage is performed first. This stage takes all columns into account; however it first indexes them into their reversed order before any are considered. This order reversal allows the columns with higher index values to be the most likely columns to be assigned. The reduction transfer stage exists solely to further reduce the amount of unassigned rows before beginning the augmenting reduction stage. The augmenting reduction stage is designed to find "augmenting paths that begin in unassigned rows and where reduction is transferred" (Jackson *et al,* 2008).

In their research, Jonker and Volgenant (1987) warned that this augmenting reduction phase can be more time consuming than the traditional methods of column and row reduction, the augmenting reduction method allows the solution to approach the optimal solution value much more rapidly and therefore be more rewarding to the algorithm in the long run (Jonker and Volgenant, 1987).

There is a possibility (albeit a small one) that after the Initialization phase, all of the necessary assignments will be made. If this is the case, the Termination phase goes into effect which terminates the algorithm as it will have already found its solution in the first phase. If this is not the case, the algorithm will move into the next phase.

The next phase of the SAP algorithm is the augmentation phase. During this phase a modified version of Dijkstra's algorithm is utilized in order to find the shortest augmenting path. This modification allows the algorithm to find the shortest augmenting path for one additional solution at the root node of the shortest path tree found by Dijkstr'a algorithm. During this phase, full or partial solutions are found using alternating rows and columns within the algorithm.

The last phase of the SAP algorithm is the adjustment of the dual solution. This phase allows the information from the initialization phase and the augmentation phase to be adjusted so that the dual variables are updated. "This allows for the restoration of complementary slackness and causes all assignments to correspond to the row minima from the reduced cost matrix." (Jackson *et al,* 2008).

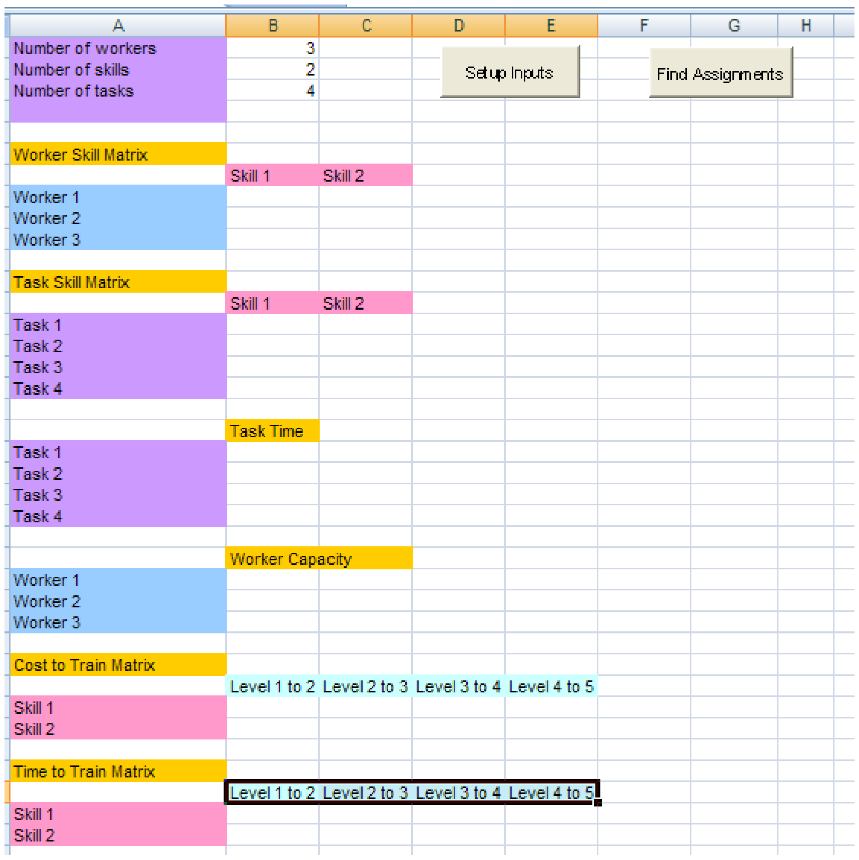
Again, it is important to emphasize the fact that this SAP method is only applicable for one-to-one applications (where the number of tasks is equal to the number of workers). Because this research is not restricted to the one-to-one case, the SAP algorithm will only be incorporated in phase 1 of the assignment. The assignment of any remaining tasks (i.e. those tasks not considered by the phase 1 SAP procedure) will be assigned using the phase 2 method previously discussed and shown in Figure 4. 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,
                worker_capacity, total_training_cost
Loop
```

FIGURE 5 – Pseudocode for Shortest Augmenting Path Algorithm (Phase 1 Only)

4. Software Tool

The three solution methodologies developed in the previous chapter were coded as Visual Basic Macros in Microsoft Excel® and an automated skills management tool was developed. There is an initial macro which is used to set up a new assignment problem. This macro prompts the user to input the necessary assignment problem data parameters such as number of workers, number of skills, number of tasks, current skill levels for each worker, required skill levels for each task, time to complete each task (after all training has been completed), each worker's capacity, training times for each skill, and training costs for each skill. Figure 6 below shows a screenshot of these empty matrices set up for a very small sized problem for 3 workers, 2 skills, and 4 tasks.



A	B	C	D	E	F	G	H
Number of workers	3						
Number of skills	2						
Number of tasks	4						
Worker Skill Matrix							
	Skill 1	Skill 2					
Worker 1							
Worker 2							
Worker 3							
Task Skill Matrix							
	Skill 1	Skill 2					
Task 1							
Task 2							
Task 3							
Task 4							
	Task Time						
Task 1							
Task 2							
Task 3							
Task 4							
	Worker Capacity						
Worker 1							
Worker 2							
Worker 3							
Cost to Train Matrix							
	Level 1 to 2	Level 2 to 3	Level 3 to 4	Level 4 to 5			
Skill 1							
Skill 2							
Time to Train Matrix							
	Level 1 to 2	Level 2 to 3	Level 3 to 4	Level 4 to 5			
Skill 1							
Skill 2							

FIGURE 6: Empty Matrices Screen Shot

V. RESULTS

Several data sets were randomly generated and are used to compare the results of the three solution methodologies discussed in the previous chapter; 1. Greedy Assignment Algorithm, 2. Meta-RaPS Greedy Assignment (MR Greedy) and 3. Meta-RaPS Shortest Augmenting Path (MR SAP). First four small data sets were generated to compare the results of the three solution methodologies to the optimal solutions. As discussed in DePuy *et al.* (2006) optimal results for small problems can be obtained using LINGO software and the mathematical model presented in Chapter II. However the optimal solution for larger problems cannot be found in a reasonable time therefore motivating the development of the solution heuristics developed in this research. Table I shows the deviation from the optimal objective function value for each of the three solution methodologies.

TABLE I

RESULTS FOR SMALL DATA SETS

Problem Size (#Workers, #Tasks, #Skills)	Optimal	Solution Methodology: **Greedy**	%Diff Opt.	Solution Methodology: **MR Greedy**	%Diff Opt.	Solution Methodology: **MR SAP**	%Diff Opt.
9, 13, 11	551	605	9.80%	558	1.27%	551	0.00%
9, 13, 11	297	370	24.58%	316	6.40%	297	0.00%
9, 13, 11	393	443	12.72%	409	4.07%	393	0.00%
13, 19, 15	976	1063	8.91%	1006	3.07%	984	0.82%
		average	14.00%	average	3.70%	average	0.20%
		# opt	0	# opt	0	# opt	3

Table I shows that for small problem sizes, the greedy methodology performs the worst of the three. Not only does the greedy version not obtain any of the optimal solutions, its percent difference from optimal averages out to be 14.00%. MR greedy does much better than the purely greedy algorithm with an average percent difference of 3.7% from the optimal value although it too receives none of the optimal solutions. MR SAP is the methodology that truly shines with these small problem sizes. Its percent difference from optimal is a mere 0.20% average. Also, the MR SAP version is able to obtain three optimal solutions out of the four problem sets. It can be easily concluded that the addition of the Shortest Augmenting Path Algorithm contributes substantially to the optimality of the solution value.

Next the three solution methodologies were evaluated using larger data sets. Table II shows results for 15 medium and 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 a commercial solution.

TABLE II

RESULTS FOR MEDIUM AND LARGE DATA SETS

Problem Size (#Workers, #Tasks, #Skills)	Phase 1: Greedy Phase 2: Greedy	Phase 1: MR Greedy Phase 2: MR Greedy	Phase 1: SAP Phase 2: Greedy	Phase 1: MR SAP Phase 2: MR Greedy
50, 55, 50	15423	15241	14922	14760
50, 75, 50	16435	15717	16089	15394
50, 100, 50	20659	19743	20600	19436
100, 110, 50	29428	29205	28468	28172
100, 150, 50	30834	30073	30089	29452
100, 200, 50	36138	35198	35413	34501
200, 220, 50	56944	56356	54248	53875
200, 300, 50	60810	59502	59045	58055
200, 400, 50	71835	69939	71254	69362
500, 550, 50	140688	140001	134598	134018
500, 750, 50	148238	146302	144220	141690
1000, 1100, 50	274574	274211	264631	263256
1000, 1500, 50	288939	286342	281221	279387
2000, 2200, 50	541494	541096	521070	519659
2000, 3000, 50	565549	562565	549748	547275

Table II shows that even with large sized problems, the MR SAP method still maintains its dominance over the other methods. This can be seen in the results for each and every large data set tested. The MR SAP version attains solution values much lower than those of the other methods. When comparing each methodology to the MR SAP method, it can be seen that the MR SAP attains values averaging 4.69% lower than those of the purely Greedy, 2.93% lower than the MR Greedy and 1.78% lower than the SAP Greedy. This is a great improvement for the tool and allows for great potential savings.

To investigate the performance of the solution methodologies for various ratios of workers and tasks, two sets of data were generated (one set with 8 problems and the other set with 10 problems) each with varied ratios of workers and tasks. Table III shows the results for these data sets. Again, due to problem size, optimal solutions are not available.

TABLE III

RESULTS FOR DATA SETS WITH VARIED RATIO OF TASKS TO WORKERS

Problem Size (#Workers, #Tasks, #Skills)	Solution Methodology: **Greedy**	Solution Methodology: **MR Greedy**	Solution Methodology: **MR SAP**
9, 9, 11	427	395	391
9, 12, 11	562	533	527
9, 15, 11	690	629	623
9, 17, 11	740	659	658
9, 18, 11	756	705	690
9, 21, 11	884	731	719
9, 27, 11	1252	899	857
9, 36, 11	1079	926	887
11, 11, 13	685	632	621
11, 14, 13	735	702	686
11, 16, 13	806	751	723
11, 18, 13	849	812	772
11, 19, 13	887	849	834
11, 21, 13	961	890	874
11, 22, 13	1022	913	908
11, 28, 13	1283	1114	1084
11, 33, 13	1600	1290	1271
11, 44, 13	1743	1462	1405

Table III shows again the strength that the SAP algorithm adds to the MR SAP heuristic. The MR SAP version outperforms the other two for each of the large data sets. It is important to notice that that difference in the solution values are not always very large. As the problem size grows for these data sets, the MR SAP has a tendency to perform slightly worse than it did on smaller problem sizes. This is due to the use of the

Phase 2. Although Phase 1 will always receive the optimal value for the worker-task combinations under consideration, as the problem size grows, a larger majority of the worker-task assignments are made during the second phase which does not guarantee an optimal solution. From this we can see that as the ratio of tasks to workers grows the solution values stray farther from the optimal values. Although this is the case, the MR SAP method is still very promising for completing these assignments with minimal costs.

As mentioned in Chapters II and IV, the original skills management problem can be slightly modified to include the option of fixed assignments. Table IV shows the results for a data set with 9 workers, 13 tasks, and 11 skills. Each row of Table IV shows the results for an increasing number of fixed assignments. The optimal results were obtained for each problem instance, and the results for each solution methodology are compared to optimal in Table IV. Obviously, as more task assignments are fixed, the problem becomes easier to solve since there are fewer 'free' or unfixed assignments that need to be considered by the solution methodologies.

TABLE IV

RESULTS FOR FIXED ASSIGNMENT DATA SETS

# fixed assignments	Optimal	Solution Methodology: **Greedy**	%Diff Opt.	Solution Methodology: **MR Greedy**	%Diff Opt.	Solution Methodology: **MR SAP**	%Diff Opt.
0	566	617	9.01%	570	0.71%	566	0.00%
1	585	646	10.43%	595	1.71%	585	0.00%
2	595	639	7.39%	603	1.34%	597	0.34%
3	606	639	5.45%	608	0.33%	606	0.00%
4	606	644	6.27%	608	0.33%	606	0.00%
5	612	644	5.23%	632	3.27%	612	0.00%
6	616	651	5.68%	638	3.57%	616	0.00%
7	616	651	5.68%	638	3.57%	616	0.00%
8	648	651	0.46%	648	0.00%	648	0.00%
9	648	648	0.00%	648	0.00%	648	0.00%
		average	5.56%	average	1.48%	average	0.03%
		# opt	1	# opt	2	# opt	9

The addition of fixed assignments adds to the complexity of assigning workers to tasks. Table IV shows how each of the methodologies performed for ten difference data sets ranging from no fixed assignments to all workers receiving one fixed assignment. Although these are small sized data sets, they can show us a lot about the capabilities of the three methods. The Greedy method was able to find only one optimal solution which was for the data set with the most fixed assignments. In general, as the number of fixed assignments grew, the Greedy heuristic was able to obtain more improved results. This can be attributed to the fact that it had fewer decisions to be made with allowed it to refrain from becoming stuck at local optima.

The MR Greedy method performed better than the purely greedy heuristic. This method was able to obtain two optimal solutions, but again these were found for data sets which included a large number of fixed assignments. The MR Greedy version averaged only 1.48% difference from the optimal solution which was also much better than the 5.56% difference obtained by the greedy version. It is easy to see that the addition of the MR heuristic is a good addition to this tool.

Overall, the MR SAP version far outperforms the other two methods once again for these data sets. This method was able to find nine optimal values out of the ten data sets. This gave it an average percent difference of 0.03% from optimal. The small size of the data sets allows most of the assignments to be made in the SAP portion of the heuristic. This gives the MR SAP method a huge advantage over the other two and proves that it is a wonderful addition to this tool.

Computer run times for these solution methodologies are obviously a function of the problem size. Run times for 10,000 iterations of MR Greedy and MR SAP for a problem of size 9 workers – 17 tasks - 11 skills was 16.14 seconds and 16.43 seconds respectively on a Dell Inspiron I6400 PC with 1.00 GB of RAM. For larger problems of size 11 workers – 17 tasks - 13 skills, it took 25.53 seconds and 30.53 seconds respectively to run 10,000 iterations. The Greedy algorithm arrives at the same solution each time it is executed and therefore only needs to be run for one iteration for each problem. Run times for the Greedy heuristic ranged from 0.15625 seconds for a small problem of size 9 workers – 17 tasks - 11 skills to 0.17188 seconds for a large problem of size 11 workers – 17 tasks - 13 skills.

VI. CONCLUSIONS AND FUTURE RESEARCH

This research compared three solution methodologies for the skills management problem of assigning tasks to workers. A Greedy Assignment algorithm was developed and two Meta-RaPS meta-heuristic approaches were developed; one based on the Greedy Assignment algorithm, the other based on the Shortest Augmented Path (SAP) algorithm of Jonker and Volgenant (1987). All three solution heuristics were included in software tool and the performance of the heuristics was evaluated using several randomly generated data sets. The Meta-RaPS SAP heuristic was shown to provide the best results.

As shown in Chapter V, the results of all data sets tested were much improved for the Meta-RaPS SAP heuristic. The improvement provided by the addition of the SAP algorithm is perhaps easiest to see in the results from the small data sets contained in Table I. This is due to the fact that the first phase of the heuristic is able to make a much larger impact as the number of assignments made in the first phase is greater when the ratio of tasks to workers is closest to one.

As the problem sizes increase, as shown in Tables II and III, the solution values grow as well. This is due to the fact that the ratio of tasks to workers is straying farther away from a value of one causing an increasing number of assignments to be made in phase 2 of the heuristic. Although phase 2, which uses the MR Greedy method, is much improved over the purely Greedy method, it does not guarantee an optimal solution as in phase 1. Even with the larger problem sizes, the MR SAP heuristic steadily maintains solution values better than those of the Greedy or the MR Greedy heuristics.

The addition of fixed assignments causes another hurdle for the heuristic tool to conquer. Table IV shows results for data sets of small to medium size that include fixed assignments. As seen from these results, the Greedy and MR Greedy heuristics continue to have difficulty in reaching optimality. The MR SAP heuristic does exceedingly well under these conditions. The fixed assignments no longer need to be considered by either part of the tool which allows the SAP portion of the heuristic to make a larger percentage of the assignments than it normally would for that size problem. This results in less assignments being made by the second phase, increasing the overall optimality of the solution.

Because the SAP algorithm was shown to be so beneficial in solving these skills management problems and because the current phase 2 algorithm has been shown to need improvement, it is recommended that future research be conducted in order to explore the capabilities of the SAP algorithm's use in Phase 2 of this skills management tool. Also, the solution methodologies should be evaluated on additional large problems.

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.

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

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.

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.

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.

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

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

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.

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.

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.

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(1996): 600-610.

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

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

Moreno, N., Corominas, A., 2007. Solving the minsum product rate variation problem as an assignment problem, *Int. J. Flexible Manufacturing Systems,* 18: 269-284.

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.

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,” *INFOR*, 45 (3): 123-141.

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.

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

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

APPENDIX I

USER'S GUIDE

Skills Management Heuristic:

A User's Guide

Prepared by: Erin Jackson

Table of Contents:

Chapter One: Introduction

1.1 Overview

Workforce competency assessment and the resulting assignment of workers to tasks is a necessary management function in most organizations. The ability to utilize current worker skills and decrease training costs is an important tool in industry. When done properly, it allows all tasks to be assigned to workers who are qualified and minimizes the total cost of training over all employees, thus minimizing the cost to the company.

This management heuristic compiles all worker skill data that is provided by the user, and calculats the optimal or near optimal solution, depending on the number of workers and the number of tasks. The tool considers the skill sets of each worker and enumerates the costs for each worker-task pairing. After passing through much iteration the tool then settles on the best overall solution.

The first phase of the tool utilizes the Shortest Augmenting Path algorithm developed by Jonker and Volgenant [1]. This portion of the tool has the ability to obtain the optimal solution however, it is only able to complete one-to-one assignments. A one-to-one assignment refers to circumstances where the number of workers is equal to the number of tasks to be assigned. For cases where the number of tasks is not equal to the number of workers, all unassigned tasks are sent to the second phase of the operation.

Phase 2 of the management heuristic contains a randomized meta-heuristic known as Meta-RaPS which was developed by DePuy, Whitehouse, and Moraga [2]. This heuristic is based on a greedy algorithm which has been randomized in order to keep the solution away from local optima. This phase of the heuristic does not guarantee and optimal solution although it has been proven to be very effective.

This tool also has the ability to complete fixed assignments. This allows the user to designate certain workers to complete specific tasks as needed.

1.2 Heuristic Tool Overview

The following factors are needed to successfully run the Logistics Skill Management Heuristic.

<u>Set-up Parameters:</u>

- Number of workers
- Number of possible skills
- Number of tasks
- Number of fixed assignments

<u>Input Parameters:</u>

- The skill level (1 -5) for EACH worker corresponding to EACH skill
- The skill level (1 -5) required for each skill that is to be assigned
- The time it takes to complete each task
- The capacity of time that each worker has available
- The cost to train between each skill level for each skill needed
- The time it takes to train a worker to the next skill level for each skill

<u>Results:</u>

The resulting data that is provided by this tool is as follows:

- Results Summary:
    - Worker – task assignment pairs
    - The cost of the best solution found
    - Run time of the model

- Worker Training Results:
    - List of each worker that required training with the skill in which they need trained, along with their original level of that skill and the skill level they need to obtain.
    - List of all workers and the number of skills they require training in (if any)
    - Training cost of each worker
    - Training time for each worker

- Skill Training Results:
    - The number of workers that need to be trained from level a to level b for each skill that requires some training.
- Totals:
    - The total number of all workers who need training from level a to level b. (Also can show this in graphical form)
    - The total overall training time
    - Total overall training cost
    - Total cost of the solution

Chapter Two: Installation

2.1 Systems Requirements

The Logistics Skills Management Heuristic can be run from Excel 2003 or Excel 2007.

2.2 Installation Steps

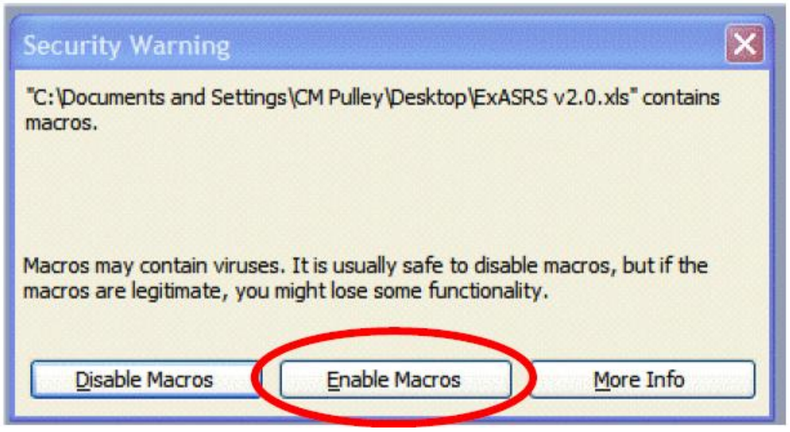
The Logistics Skills Management Heuristic is run in a usual Excel workbook.

Step 1: Download the file (ExcelProject.xls) and save to desired location.

Step 2: Double-click on the saved file to open.

Step 3: Enable Macros:

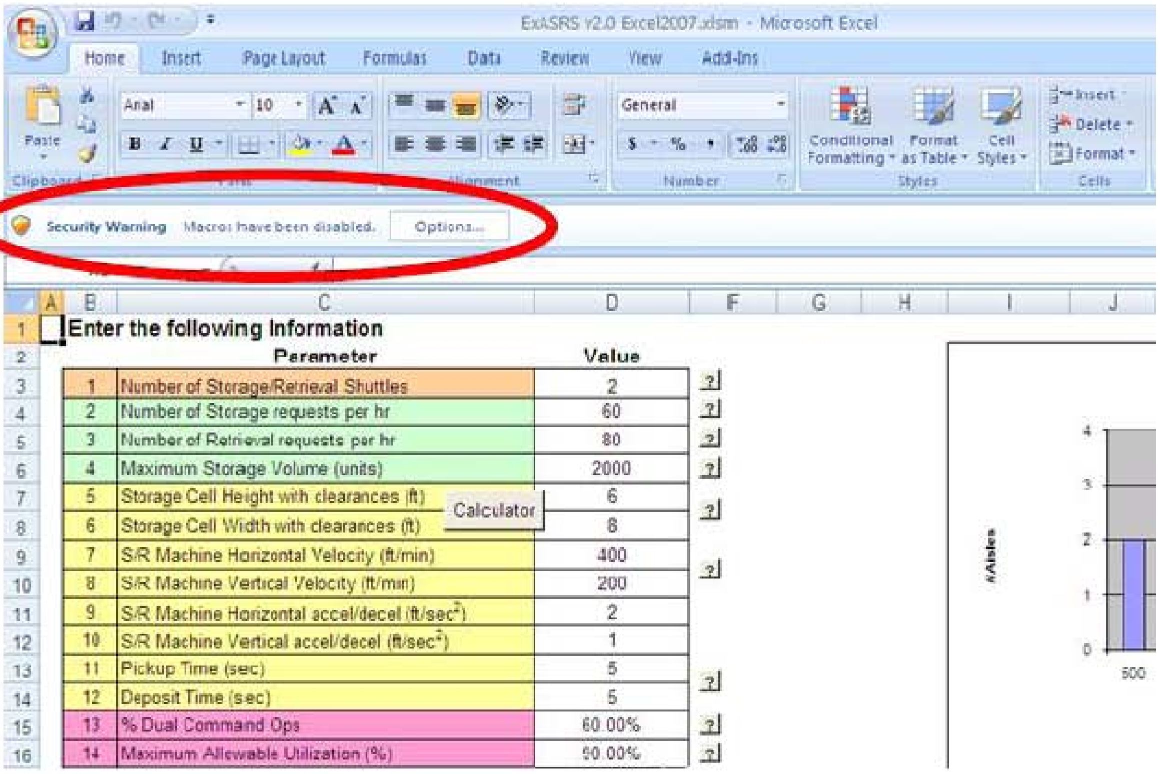
- For a 2003 workbook the user will be prompted with a security warning. Click "Enable Macros" in order to run the tool. [3]



"C:\Documents and Settings\CM Pulley\Desktop\ExASRS v2.0.xls" contains macros.

Macros may contain viruses. It is usually safe to disable macros, but if the macros are legitimate, you might lose some functionality.

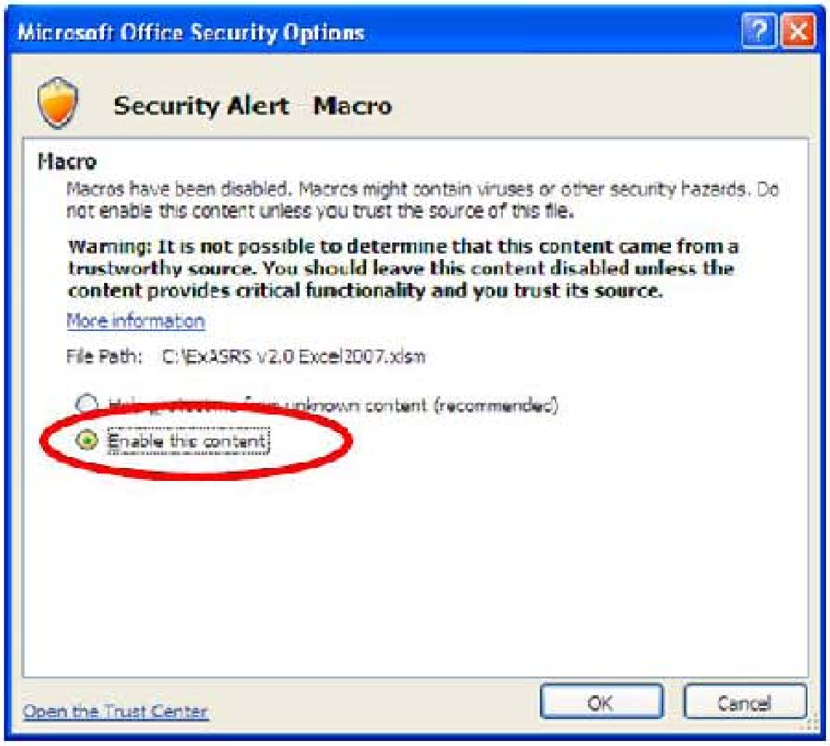
- For a 2007 workbook the security warning is shown towards the top of the page. Click "Options" then "Enable this Content". [3]



Security Warning Macros have been disabled.

Enter the following Information

	Parameter	Value
1	Number of Storage/Retrieval Shuttles	2
2	Number of Storage requests per hr	60
3	Number of Retrieval requests per hr	80
4	Maximum Storage Volume (units)	2000
5	Storage Cell Height with clearances (ft)	6
6	Storage Cell Width with clearances (ft)	8
7	S/R Machine Horizontal Velocity (ft/min)	400
8	S/R Machine Vertical Velocity (ft/min)	200
9	S/R Machine Horizontal accel/decel (ft/sec^2)	2
10	S/R Machine Vertical accel/decel (ft/sec^2)	1
11	Pickup Time (sec)	5
12	Deposit Time (sec)	5
13	% Dual Command Ops	60.00%
14	Maximum Allowable Utilization (%)	90.00%



Security Alert - Macro

Macro

Macros have been disabled. Macros might contain viruses or other security hazards. Do not enable this content unless you trust the source of this file.

Warning: It is not possible to determine that this content came from a trustworthy source. You should leave this content disabled unless the content provides critical functionality and you trust its source.

More information

File Path: C:\ExASRS v2.0 Excel2007.xlsm

Help protect me from unknown content (recommended)

[✓ Enable this content]

Chapter Three: Tool Operations

3.1 Running the Assignment Tool

1. Open the Excel workbook (ExcelProject.xls) by double clicking on the file

2. Enable Macros (as instructed in Chapter 2)

3. Click the button that reads "Setup Inputs"

4. When prompted, enter the desired number of workers, skills, tasks and fixed assignments.

5. After the sheet has completed its setup, enter the necessary parameters (See Input Parameters listed in Chapter 1)

6. Once all parameters are entered, click the button that reads "Find Assignments"

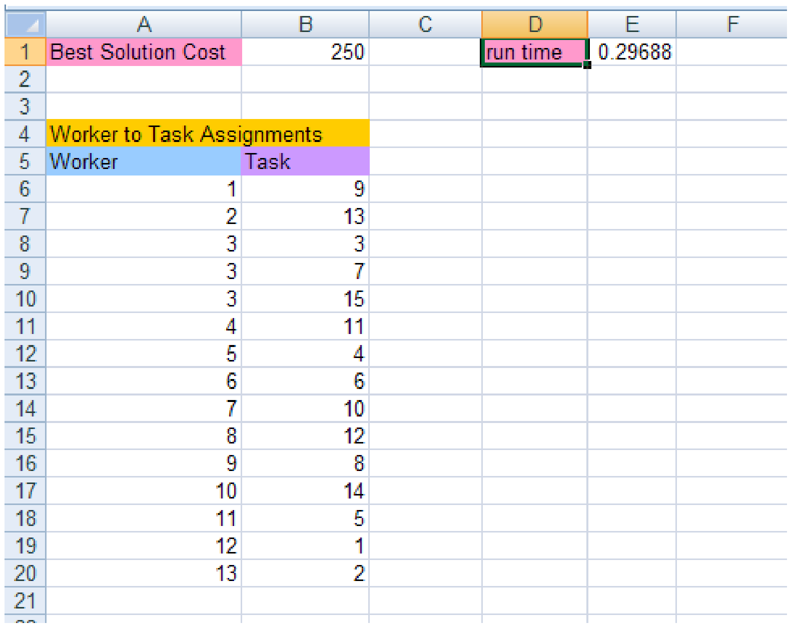
This tool will not save multiple runs of an assignment within its workbook. For this reason it may be a good idea to save each assignment as a different file. To do this, select the file menu at the top of the workbook (in Excel 2007 this appears as the Microsoft Office Logo at the top left of the screen) and select "Save As". Rename the workbook to the title of your choice and click "Save".

As a user of this workbook, that tab labeled 'Parameters' may also be of interest. This tab contains the percentage parameters and number of iterations that are used in the heuristic code for the assignment calculations. This number, when changed, will cause the second phase of the operation to find a different solution. Use this feature cautiously as it could result in a solution that is farther from the optimal; however it can be changed easily and may prove helpful. To change any of these parameters, simple click the arrow on the drop-down box and select the number of your choice, then rerun the tool by clicking the "Find Assignments" button on the 'Input' tab.

Chapter Four: Results

4.1 Basic Results

The "Results Summary" page contains a quick overview of the assignment solution. This page includes the following information: Each worker-task pair that is assigned, the cost of this solution, and the run time for the run. This data is presented in the format shown below.

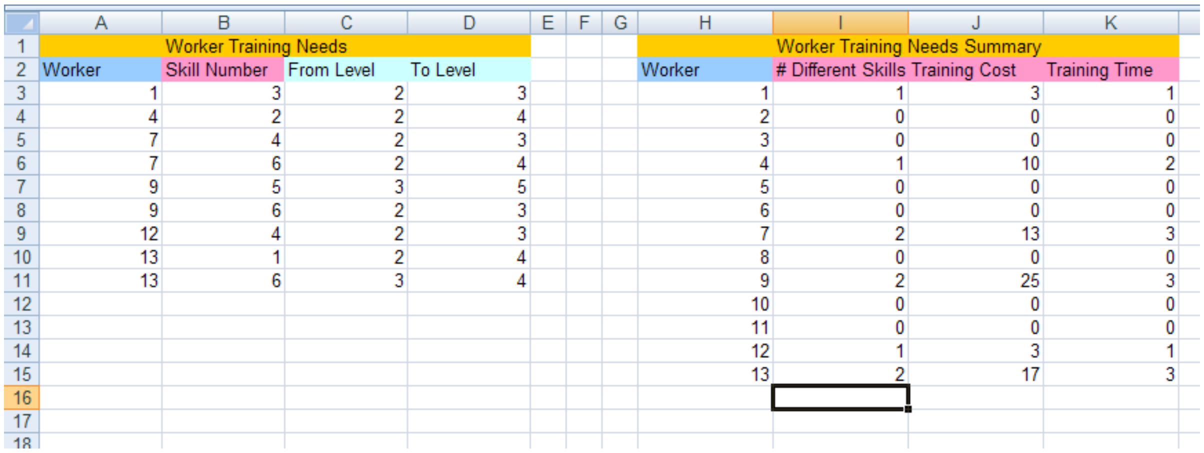


	A	B	C	D	E	F
1	Best Solution Cost	250		run time	0.29688	
2						
3						
4	Worker to Task Assignments					
5	Worker	Task				
6	1	9				
7	2	13				
8	3	3				
9	3	7				
10	3	15				
11	4	11				
12	5	4				
13	6	6				
14	7	10				
15	8	12				
16	9	8				
17	10	14				
18	11	5				
19	12	1				
20	13	2				
21						

4.2 Additional Results

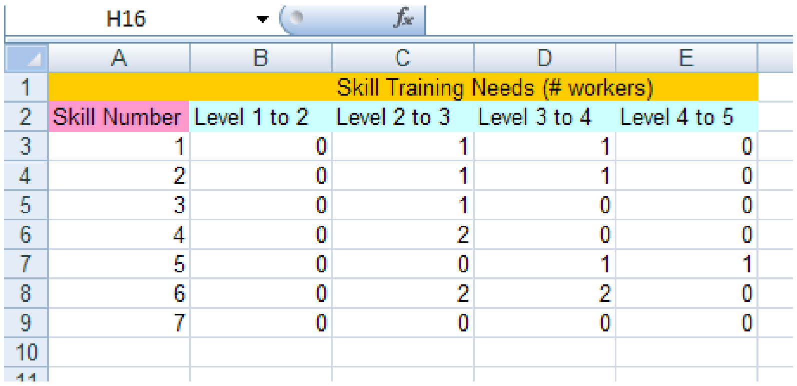
Additional results are displayed in various other tabs of the workbook.

1. To view detailed results for the training required by each worker, click on the 'Worker Training Results' tab. This worksheet provides information on which workers need to be trained on which tasks. It also shows what skill levels that begin at for these tasks as well as what skill levels they need to be trained to. Another table shows all of the workers in the assignment and how many (if any) different skills they need to be trained on, along with the training costs and training times associated with each of these workers. An example of this sheet is shown below.



	A	B	C	D	E	F	G	H	I	J	K
1	Worker Training Needs							Worker Training Needs Summary			
2	Worker	Skill Number	From Level	To Level				Worker	# Different Skills	Training Cost	Training Time
3	1	3	2	3				1	1	3	1
4	4	2	2	4				2	0	0	0
5	7	4	2	3				3	0	0	0
6	7	6	2	4				4	1	10	2
7	9	5	3	5				5	0	0	0
8	9	6	2	3				6	0	0	0
9	12	4	2	3				7	2	13	3
10	13	1	2	4				8	0	0	0
11	13	6	3	4				9	2	25	3
12								10	0	0	0
13								11	0	0	0
14								12	1	3	1
15								13	2	17	3
16											
17											
18											

2. To view additional results regarding skills and skill levels for training, click on the tab labeled ‘Skill Training Results’. This worksheet lists all of the skills that workers will need to be trained on. It also shows the number of workers that need to be trained on each of these skills and at what level they will need training. An example of this output is shown below.



H16

	A	B	C	D	E
1	Skill Training Needs (# workers)				
2	Skill Number	Level 1 to 2	Level 2 to 3	Level 3 to 4	Level 4 to 5
3	1	0	1	1	0
4	2	0	1	1	0
5	3	0	1	0	0
6	4	0	2	0	0
7	5	0	0	1	1
8	6	0	2	2	0
9	7	0	0	0	0
10					
11					

3. As a user, if it is desired to view some overall totals of some of these results, right click on the last tab and select ‘Unhide’. A box will pop-up with a sheet named ‘Totals’. Select this sheet and click ‘OK’. This will bring up the hidden sheet that contains some basic sums of this information. This information is also formatted to show any non-zero training values in green for easy interpretation. Also, a

button is included on this sheet that when clicked, will produce a graph of the training summed information. When use of this tab is complete, right click on the ‘Totals’ tab and select ‘Hide’ to restore the hidden quality to the tab.

VITA

Erin Lynn Jackson

eljack04@louisville.edu

EDUCATION

Masters of Engineering in Industrial Engineering……………………………….May 2009
University of Louisville, Louisville, KY

Bachelors of Science in Industrial Engineering………………………………August 2008
University of Louisville, Louisville, KY

HONORS AND AWARDS

Speed School Alumni Scholarship……………………………………………August 2004
University of Louisville Honors Program……………………August 2004 – August 2008
MHEFI Scholarship………………………………………………………..August 2008
GEMS Mathematics Fellowship…………………………………August 2008 - May2009

EXTRACURRICULAR ACTIVITIES

Executive Board member Golden Key International Honour Society
Member of Society of Women Engineers U of L Chapter
Member of National Society of Collegiate Scholars
Member of Institute of Industrial Engineers
Member of Tau Beta Pi Engineering Honor Society