The $0-1$ Multiple Knapsack Problem

by

Hayat Abdullah Shamakhai

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc) in Computational Science

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	The 0-1 Multiple Knapsack Problem	
Name of Candidate Nom du candidat	Shamakhai, Hayat	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance May 31, 2017

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Youssou Gningue
(Co-Supervisor/Co-directeur(trice) de thèse)

Dr. Hafida Boudjellaba
(Co-Supervisor/Co-directeur(trice) de thèse)

Dr. Haibin Zhu
(Committee member/Membre du comité)

Dr. Matthias Takouda
(Committee member/Membre du comité)

Dr. Oumar Mandione Guèye
(External Examiner/Examinateur externe)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. David Lesbarrères
Monsieur David Lesbarrères
Dean, Faculty of Graduate Studies
Doyen, Faculté des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

In operation research, the Multiple Knapsack Problem (MKP) is classified as a combinatorial optimization problem. It is a particular case of the Generalized Assignment Problem. The MKP has been applied to many applications in naval as well as financial management. There are several methods to solve the Knapsack Problem (KP) and Multiple Knapsack Problem (MKP); in particular the Bound and Bound Algorithm (B&B). The bound and bound method is a modification of the Branch and Bound Algorithm which is defined as a particular tree-search technique for the integer linear programming. It has been used to obtain an optimal solution. In this research, we provide a new approach called the Adapted Transportation Algorithm (ATA) to solve the KP and MKP. The solution results of these methods are presented in this thesis. The Adapted Transportation Algorithm is applied to solve the Multiple Knapsack Problem where the unit profit of the items is dependent on the knapsack. In addition, we will show the link between the Multiple Knapsack Problem (MKP) and the multiple Assignment Problem (MAP). These results open a new field of research in order to solve KP and MKP by using the algorithms developed in transportation.

Keywords:
Generalized Assignment Problem, Assignment Problem, Knapsack problem, Multiple Knapsack Problem, Branch and Bound Algorithm, Bound and Bound Algorithm, Transportation Problem, Multiple Assignment Problem, Adapted Transportation Problem, Vogel Approximation Method, Group Role Assignment problem.

Acknowledgements

All praise and thanks to Allah who made all the things possible and without his blessing and guidance I'm not able to finish this thesis.

Then I would like to express my deepest gratitude and appreciation to my supervisor Dr. Youssou Gningue who supports me throughout my thesis with his patience and knowledge. His support, guidance, encouragement and suggestions helped me through work and written of this thesis.

Also, I would like to extend my thanks to Dr. Hafida Boudjellaba to be my co-supervisor and to my committee members Dr. Matthias Takouda and Dr. Haibin Zhu.

I am indebted to my lovely dad Abdullah and my lovely mom Mariam; I am where I am today because of your love, support, prayers and encouragements.

Of course special thanks go to my sisters and brothers especially my dear brother Hassan who has always supported me.

Also, I would like to thank my government of Saudi Arabia for giving this opportunity to complete my master degree.

Table of Contents

List of Figures

List of Tables

Abbreviations

GAP	Generalized Assignment Problem
VDSH	Variable Depth Search Heuristic
HH	Hybrid Heuristic
EC	Ejection Chain
MSPEC	Multi – Start Parallel Ejection Chain
CPEC	Cooperative Parallel Ejection Chain
CGA	Constructive Genetic Algorithm
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search
AP	Assignment Problem
TP	Transportation Problem
MRGAP	Multiple Resource Generalized Assignment Problem
MGAP	Multiple Generalized Assignment Problem
DGAP	Dynamic Generalized Assignment Problem
BGAP	Bottleneck Generalized Assignment Problem

GAPSOS	Generalized Assignment Problem with Special Ordered Set
SGAP	Stochastic Generalized Assignment Problem
BIGAP	Bi-Objective Generalized Assignment Problem
GMAP	Generalized Multi – Assignment Problem
MKP	Multiple Knapsack Problem
KP	Knapsack Problem
B&B	Branch and Bound Algorithm
B&B	Bound and Bound Algorithm
MTM	Martello and Toth Method
MAP	Multiple Assignment Problem
GRAP	Group Role Assignment Problem
LTP	Linear Transportation Problem
VAM	Vogel Approximation Method
ATA	Adapted Transportation Algorithm
BKP	Balanced Knapsack Problem
BMKP	Balanced Multiple Knapsack Problem
MIN – BKP	Minimization Balanced Knapsack Problem

MIN – BMKP	Minimization Balanced Multiple Knapsack Problem

Introduction

Primarily, our research interest was the Generalized Assignment Problem (GAP). The GAP is a generalization form of the classic assignment problem where a task can be assigned to more than one agent. It consists in assigning n tasks $j = 1, \dots, n$ to m agents $i = 1, \dots, m$. A task j can be performed by agent i with weight $w_{i,j}$ such that the sum of weight will not exceed the capacity W_i. The assigned of task j to agent i induces a cost $P_{i,j}$. If we consider the decision variable

$$X_{i,j} = \begin{cases} 1 & \text{if task } i \text{ is assigned to } j \\ 0 & \text{else} \end{cases}$$

The GAP can formulated as

$$GAP \begin{cases} \min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} X_{i,j} \\ \sum_{j=1}^{n} w_{i,j} X_{i,j} \leq W_i ; \quad i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} = 1 ; \quad j = 1, \dots, n \\ X_{i,j} \in \{0, 1\}; \quad i = 1, \dots, m ; \; j = 1, \dots, n \end{cases}$$

The Generalized Assignment Problem is extensively studied, and many algorithms have been proposed to solve it. The subject constitutes my first seminar presentation. This has provided to me the opportunity to make a general review on the subject. From this, we notice that the problem has a large range and collection a specific subject is very difficult of the master thesis. Therefore, we decided to narrow the research on the GAP where the

parameters $w_{i,j}$ are uniform i.e. $w_{i,j} = 1$ and called Uniform Generalized Assignment Problem (UGAP). The UGAP is formulated as

$$UGAP \begin{cases} \min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} X_{i,j} \\ \sum_{j=1}^{n} X_{i,j} \leq W_i; \;\; i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} = 1; \;\; j = 1, \dots, n \\ X_{i,j} \in \{0,1\}; \;\; i = 1, \dots, m \; ; \;\; j = 1, \dots, n \end{cases}$$

Our goal was to select some algorithms developed for the GAP in order to adapt it and simplify it for the UGAP. We notice that the UGAP was fully studied when the parameters W_i are integers as the Group Role Assignment Problem (GRAP). Zhu et al [76] had provided a very elegant approach to solve those problems. Therefore, we decided to extend the research to a more general form where $w_{i,j} = w_j$ and the coefficients of the objective $P_{i,j} = P_j$. This provides the $0-1$ minimization Multiple Knapsack Problem. The minimization MKP is a special case of the GAP where the profit and weight for each item are independent of the knapsack. It can be formulated as an integer linear programming problem

$$MKP \begin{cases} \min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_j X_{i,j} \\ \sum_{j=1}^{n} w_j X_{i,j} \leq W_i; \quad i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} = 1; \;\; j = 1, \dots, n \\ X_{i,j} \in \{0,1\}; i = 1, \dots, m \; ; j = 1, \dots, n \end{cases}$$

The MKP is a generalization of the $0-1$ Knapsack Problem where there is a single knapsack. The purpose is to provide a new approach for solving the MKP and its single form. The idea of the approach begins by linking the KP and the Linear Transportation Problem (LTP). After that, we solve the problem by using the Adapted Transportation Algorithm (ATA). We also apply this algorithm for solving the MKP. The initial solutions are obtained by using Vogel Approximation Method (VAM) which is a heuristic method of solving the Transportation Problem (TP). After that, we improve the initial solutions by using the dual variable and resulting reduced cost. The outline of this thesis is organized as following:

In chapter 1, we present the Generalized Assignment Problem because it is a generalization of the subject of this thesis. It is formulated as an integer linear programming. In this chapter, we are also presented some applications and extensions of the GAP.

In next chapter, we propose the $0-1$ Knapsack Problem and also formulate it as an integer linear programming. In addition, we focus on the Branch and Bound Algorithm (B&B) as a method for solving the problem. We illustrate the result by an example.

The following chapter is about the $0-1$ Multiple Knapsack Problem. In addition, we introduce the Branch and Bound Algorithm and the Bound and Bound Algorithm. The Bound and Bound algorithm is a modification method of the Branch and Bound Algorithm. Therefore, we also provide an illustration example that has been solved by the Bound and Bound Algorithm to obtain an optimal solution.

In chapter 4, we begin by presenting the Transportation Problem which is also formulated as an integer linear programming problem. After that, we show the link

between the Knapsack Problem and the Linear Transportation Problem. The resulting method is an adaptation of the transportation algorithm and provides an optimal solution. We apply this algorithm to the same example that has been introduced in chapter 2.

In chapter 5, we present the Multiple Knapsack – Transportation Problem (MKTP). We apply the Adapted Transportation Algorithm for solving the problem and also provide an optimal solution. Actually, in this chapter, we present two different examples. In the first the profit of each item j are independent of knapsack i while the second it is dependent of the knapsack i.

The last chapter is about the Multiple Assignment Problem which is a generalization of the classic Assignment Problem. Therefore, we introduce and formulate the problem as an integer linear programming problem. We also present some cases of the MAP. One of these cases is called Group Role Assignment Problem. In addition, we show that the GRAP is indicated to the GAP and also can be a Multiple Knapsack Problem in particular cases.

.

Chapter 1

1 Generalized Assignment Problem

1.1 Introduction

The Generalized Assignment Problem (GAP) is a problem in combinatorial optimization. It is known as a generalization form of a classic Assignment Problem (AP) when the number of both task and agent are equal. However, for the Generalized Assignment Problem (GAP), the number of agents assigned to each task could be different from one to the other.

The objective of the Generalized Assignment problem is to minimize the obtained cost without exceeding the capacity. In addition, it has been applied to many real - life applications in governments and industries such as various routing problems and flexible manufacturing systems.

Moreover, the GAP has been solved by many algorithms some of them give an optimal solution, and others provide an approximate solution.

In this chapter, we will briefly present the Generalization Assignment Problem which is a generalization of the subject of this thesis; the Multiple Knapsack Problem (MKP).

1.2 Literature review of the Generalized Assignment Problem

The GAP is known to be an NP- complete problem and cannot be solved by a polynomial – time approximation algorithm. Therefore, there are several approximation algorithms for GAP. This is due to the fact that these algorithms address a different setting where available agent capacities are not fixed and the weighted sum of cost and available agent capacity are minimized. All of these algorithms have feasible solutions; therefore, for some of them the feasible solutions are required as an input.

Using an implicit enumerative procedure can help to obtain an optimal solution for GAP. An implicit enumerative procedure has two methods: branch and bound scheme, and branch and price scheme. The branch and bound has four procedures: an upper bounding procedure, a lower bounding procedure, a branching strategy and a searching strategy. A branch and price method is known to be similar to branch and bound method; however, the bounds can be obtained by solving the LP – relaxations of the subproblems by column generation.

Ross and Soland [61] suggested the first branch and bound algorithm for solving GAP. The GAP is considered a minimization problem; therefore, Ross and Soland [61] achieve the lower bounds by relaxing the capacity constraints. Martello and Toth [43] proposed removing the semi – assignment constraints where the problem decomposes into a series of knapsack problems. Chalmet and Gelders [10] introduced the Lagrangian relaxation algorithm of the semi – assignment constraints. Fisher, Jalikumar and Wassenhove [18]

used this technique with multipliers by heuristic adjustment methods to obtain the lower bounds in the branch and bound procedure.

Guignard and Rusenwein [23] proposed a new algorithm which is Known as a branch and bound algorithm. With an enhanced Lagrangian dual ascent procedure this algorithm effectively solves GAP with up to 500 variables. This algorithm solves a Lagrangian dual at each enumeration node and adds a surrogate constraint to Lagrangian relaxation model. Drexl [15] introduced a hybrid branch and bound /dynamic program algorithm where the upper bounds are obtained by an efficient Monte Carlo type heuristic. Nauss [51] presents a branch and bound algorithm where linear programming cuts, Lagrangian relaxation, and subgradient optimization are used for achieving good lower bounds. Furthermore, Ronen [60] who proposes feasible – solution generators with heuristic uses them to derive good upper bounds. Nauss [50] has also used similar techniques as Ronen in order to solve the elastic generalized assignment problem.

For branch and price algorithm, Savelsbergh [64] proposed first branch and price algorithm to solve the GAP. Martello and Toth [40] presented a combination of branch and price algorithm.

Nasberg [28] introduced a knapsack problem by using the combination of the algorithm to calculate the upper bound and pricing problem. Barnhart et al [4] developed the formula of GAP via applying Dantzig – Wolfe decomposition to obtain a tighter LP relaxation. A series of knapsack problem can solve the LP relaxation of reformulated problem pricing. Pigatti et al [56] introduced a branch and cut – price algorithm with a stabilization mechanism to speed up the pricing convergence. Ceselli and Righini [9]

proposed a branch and price algorithm for multilevel generalized assignment problem which is based on a decomposition and a pricing subproblem: a multiple – choice knapsack problem. Since the GAP is considered as an NP – hard problem; therefore, some instances of the GAP are computationally intractable. That reason requires finding heuristic approaches. Since heuristic approaches can be used in two fold; they are used as stand – alone algorithm to obtain a good solution within a reasonable time and attempt to achieve the upper bounds in exact solution methods, for instance, the branch – and – bound procedure. Although the variety of the heuristics is high, they mostly fall into one of the following two categories: greedy heuristics and mate – heuristics.

Klastorin [30] suggests a two stages heuristic algorithm for solving GAP. In stage one, the algorithm uses a modified subgradient algorithm to search for the optimal dual solution and in stage two, a branch and bound algorithm searches about the neighborhood of the similar solution that is found in stage one.

Cattrysse et al [8] used column generation techniques to obtain upper and lower bounds. A Column is represented as a feasible assignment of a subset of tasks to a single agent. The main problem is formulated as a set partitioning problem. New columns that have been obtained will be added to the main problem by solving a knapsack problem for each agent. A dual ascent procedure can be solved using LP relaxation of the set- partitioning problem. Martello and Toth [42] proposed greedy heuristic where it helps to assign the jobs to machines based on a desirability factor. This factor is known as the difference between the largest and second – largest weight factors. This heuristic approach uses to reduce a problem size by fixing variables to one or to zero. Lorena and Narciso [35]

developed Relaxation heuristics for maximization version of GAP where a feasible solution can be obtained by a subgradient search in a Lagrangian or surrogate relaxation.

Haddadi [24] introduces a substitution variable in a Lagrangian heuristic for GAP. This is defined as the multiplication of the problem where resulted relaxation is divided into two subproblems: the transportation problem and knapsack problem. Naricso and Lorena [49] find a good feasible solution by using relaxation multipliers with efficient constructive heuristics.

Haddadi and Ouzia [25] described a branch and bound algorithm; a standard subgradient approach that uses each node of the decision tree to solve the Lagrangian dual and find an upper bound.

There is a contribution that attempts to solve a GAP of smaller size by a new heuristic that is applied to exploit solution of the relaxed problem. Romijh and Romero Morales [59] applied the optimal value function from a probabilistic point of view and expanded a class of greedy algorithms.

Variable depth search heuristic (VDSH) introduced by Amini and Racer [1] and used to solve the GAP. VDSH is defined as a generalization of local search in which the size of the neighborhood adaptively changes to traverse a larger search space. Amini and Racer [2] expand a hybrid heuristic (HH) around the two heuristics called VDSH and HGAP. There is another hybrid approach introduced by Lourenco and Serra [38] where a MAX – MIN system (MMAS) are applied with GRASP for solving the GAP.

Yagiura et al [73] suggest a new method for solving the GAP and is called a variable depth search (VDS). The main idea of this method alternates between shift and swap moves to explore the solution space. The main goal of this method is that infeasible solutions are allowed to be considered.

Yagiura et al [72] developed VDS by incorporating branching search processes to construct the neighborhoods. Yagiura et al present appropriate choices of branching strategies, which subsequently help to develop the performance of VDS.

Lin et al [34] mention several observations on VDSH method by a sequence of computational experiments. They propose six greedy strategies for generating the initial feasible solution and created new simplified strategies. This is a positive change as the purpose is to improve phase of the method.

Osman [52] expands a hybrid heuristic that incorporates simulated annealing and tabu – search.

Yagiura et al [70] suggest a tabu – search algorithm as a new heuristic for solving the GAP, which uses an ejection chain approach where an ejection chain is defined as an embedded neighborhood construction that compounds simple move to design several complex and powerful moves. Yagiura et al [70] considered the chain to be a series of shift moves where each two successive moves share a common agent.

Yagiura et al [71] develop their previous method by introducing a new approach called a path relinking approach. A path relinking approach is a known mechanism for producing new solutions by combining two or more reference solutions.

Asahiro et al [3] improve two parallel heuristic algorithms based on the Ejection Chain local search EC presented by Yagiura et al [70]. They proposed that EC has two parallels multi – start and parallel EC (MSPEC) cooperative parallel EC (CPEC).

Diza and Fernadez [14] create a flexible tabu – search algorithm for solving the GAP. The search allowed to research about an infeasible region and adaptively modifying the objective function is the source of flexibility. The modification of the objective function happens due to the dynamic adjustment of the weight of the penalty incurred for violating feasibility. In this method with tabu – search method we can explore the infeasible region and solution is qualitatively preferred to other of its structure.

Chu and Beasley [11] improve a genetic algorithm for solving the GAP. Since genetic algorithm is defined to incorporate fitness – unfitness pair evaluation function as a representation scheme. This algorithm is used as a heuristic that helps to improve the cost and feasibility of GAP.

Feltl and Raidl [16] modify this algorithm by adding new features such us a modified selection and replacement scheme for dealing with an infeasible solution more appropriately and a heuristic mutation operator. Wilson [68] suggests another algorithm for GAP, which operates in a dual sense. This algorithm has tried to genetically restore feasibility to a set of near optimal solution.

Lorena et al [36] suggest a constructive genetic algorithm (CGA) for solving GAP. There are some new features for CGA that can be compared to GA such us dynamic population and population formation by schemata, etc...

Lourenco and Serra [37] propose two metaheuristic algorithms for solving GAP. The first is a MIN – MAX ant system and is combined with local search and tabu – search heuristics. The second metaheuristic is a greedy randomized adaptive search heuristic (GRASP), which is considered with several neighborhoods.

Monfared and Etemadi [47] apply a neural network as an approach for solving the GAP.

1.3 The Mathematical Formulation of the GAP

It consists of minimizing the cost of assigning n tasks to m agents, so each task should be assigned to only one agent, subject to a capacity constraint for each agent.

The GAP can be formulated as an integer programming problem

$$GAP \begin{cases} \min \mathrm{Z} = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} X_{i,j} \\ \sum_{j=1}^{n} w_{i,j} X_{i,j} \leq W_i; \quad i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} = 1; \quad j = 1, \dots, n \\ X_{i,j} \in \{0,1\}; \quad i = 1, \dots, m; \ j = 1, \dots, n \end{cases}$$

The coefficient $P_{i,j}$ are the cost of assigning task j to agent i, $w_{i,j}$ is the requirement coefficients "weight" when task j is assigned to agent i and W_i is the capacity that is available for agent i. The objective function represents the total cost of assigning task j to agent i. The first set of constraints represents the total weight of assigning task j to agent i does not exceed the capacity W_i. For the third equation, it says each task j is assigned to exactly one agent. The final equation represents the binary conditions on the decision

variables where $X_{i,j}$ take on the value 1 when task j is assigned to agent i and 0 otherwise.

The Generalized Assignment Problem (GAP) usually has a large dimension and is a very general problem which is constituted by some special cases of a problem.

The $0-1$ Multiple Knapsack Problem is a special case of the Generalization Assignment Problem (GAP) when item j assigns to knapsack i with weight w_j, profit p_j and capacity W_i.

The classical Assignment Problem (AP) is also a special case of the Generalization Assignment when $w_{i,j} = 1$ for all $i \in m, j \in n$ and $m = n$.

Also, the Generalized Assignment Problem (GAP) can be interpreted as a specialized Transportation Problem (TP) when the quantity demanded at each destination should be supplied by a single origin and $w_{i,j}$ is constant for each i.

In the next section, we introduce the relaxation of the Generalized Assignment Problem (GAP).

1.4 Relaxation heuristic for the Generalized Assignment Problem

In this section, we present two different types of relaxation heuristics for the GAP: Lagrangian Relaxation and Surrogate Relaxation.

1.4.1 Lagrangian Relaxation

Lagrangian relaxation has been introduced in many survey papers [61] [17] [35] and books [55] [43]. It based on the relaxation of the capacity constraints. The GAP is defined by using a positive vector $(\lambda_1, \dots, \lambda_n)$ of multipliers and is formulated as

$$L(GAP, \lambda) \begin{cases} \max Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} X_{i,j} - \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{m} X_{i,j} - 1 \right) \\ \sum_{j=1}^{n} w_{i,j} X_{i,j} \leq W_i \ ; \ \ i = 1, \dots, m \\ X_{i,j} \in \{0, 1\}; \quad i = 1, \dots, m \end{cases}$$

Martello and Toth [43] proved the Lagrangian relaxation might be decomposed in m independent $0 - 1$ knapsack problem. Therefore, the optimal solution of $L(GAP, \lambda)$ is

$$z\big(L(MKP, \lambda)\big) = \sum_{i=1}^{m} z_i + \sum_{j=1}^{n} \lambda_j$$

Lorena and Narciso [35] showed that the optimal value of the Lagrangian Relaxation which is greater than or equal to the optimal value of the maximization version of the Generalized Assignment Problem.

1.4.2 Surrogate Relaxation

Surrogate Relaxation was proposed by Glover [21]. For the Generalized Assignment Problem, it defined a positive vector $(\pi_1, \dots, \pi_n)$ of multipliers and is formulated as

$$S(GAP,\pi)\begin{cases} \max \mathrm{Z} = \sum_{i=1}^{m}\sum_{j=1}^{n} P_{ij}\, X_{i,j} \\ \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} w_{i,j}\, X_{i,j} \le \sum_{i=1}^{m} \pi_i\, W_i \\ \sum_{i=1}^{m} X_{i,j} = 1;\ \ j = 1,\dots,n \\ X_{i,j} \in \{0,1\};\ \ i = 1,\dots,m\,; j = 1,\dots,n \end{cases}$$

Lorena and Narciso [35] also present the optimal value of the Surrogate Relaxation which is greater than or equal to the optimal value of the maximization version of the GAP.

In addition, Lorena and Narciso [49] developed the previous relaxations by providing Lagrangean/surrogate relaxation for the GAP.

In the following section, we present applications and extensions of the Generalized Assignment Problem.

1.5 Generalized Assignment Problem Application and Extensions

The Generalization Assignment Problem (GAP) has been used to describe various real – world applications. Therefore, there are several extensions of the Generalization Assignment Problem (GAP) such as the Multiple Resource Generalized Assignment Problem (MRGAP), the Multilevel Generalized Assignment Problem (MGAP), the Dynamics Generalized Assignment Problem (DGASP), the Bottleneck Generalized

Assignment Problem (BGAP), the Generalized Assignment Problem with Special Ordered Set (SOS), the Stochastic Generalized Assignment Problem (SGAP), the Bi-Objective Generalized Assignment Problem (BiGAP), and the Generalized Multi-Assignment Problem (GMAP). These applications and extensions are briefly presented in order to show the diversity of the Generalization Assignment Problem. Most of these extensions can also be applied to the Multiple Knapsack Problem as a particular case of the Generalization Assignment Problem.

1.5.1 Multiple Resource Generalized Assignment Problem

Gavish and Pirkul [20] developed this assignment problem. It is considered as a special case of the multi – resources weighted assignment model and was studied by Ross and Zoltners [62]. MRGAP is defined to have a set of tasks and a set of agents, so a set of tasks should be assigned to a set of multiple resources consumed by an agent. In MRGAP each agent will consume diverse resources to perform tasks that have been assigned to the agent. MRGAP in large models can deal with processors and database locations in a distributed computer system [20]. The truck routing problem is an application that can be modeled as multi – resources weighted assignment.

1.5.2 Multilevel Generalized Assignment Problem

Glover et al [22] proposed MGAP, which is known to process a GAP where the agents can perform tasks at more than one level. The manufacturing problem is considered as an application that can be formulated as MGAPs [53].

1.5.3 Dynamic Generalized Assignment Problem

The DGAP purposes to track customer demand while assigning tasks to agents. Kogan et al. [31] have been adding the impact of time to the GAP model assuming that each task has a due date. They formulate an optimal control model for the problem where it supplies a dynamic system by analytical properties of the optimal behaviour.

1.5.4 Bottleneck Generalized Assignment Problem

The Bottleneck Generalized Assignment Problem (BGAP) occurs when the maximum of the individual cost are considered instead of the sum. Therefore, the problem becomes a minimax problem [46]. It means all maximum penalty incurred by assigning each agent to each task is minimized. It has applied to several applications such us in scheduling and allocation problems [44].

1.5.5 GAP with Special Ordered Set

The Generalized Assignment Problem (GAP) with special order set (SOS) proposed by Beale and Tomlin [5] includes some cases where each item can share via a pair of adjacent knapsacks that can call the GAP with the special ordered set (SOS). The GAP with SOS arises mainly in production scheduling and means to allocate each task to a time- period as introduced by Farias et al [13].

1.5.6 Stochastic Generalized Assignment Problem

Stochasticity can be seen in GAP because an available resource requests to process tasks by the different agents could not be known in advance or the presence or absence of individual tasks may be uncertain. In some cases, some tasks may or may not need to be processed.

1.5.7 Bi-Objective Generalized Assignment Problem

Zhang and Ong [74] considered a GAP with a multi – objective and suggested an LP-based heuristic for solving Bi-Objective Generalized Assignment Problem. It is noted that each assignment has two objectives that have been already considered. It is applied in production planning where these attributes could be the cost and the time caused by assigning jobs to tasks.

1.5.8 Generalized Multi-Assignment Problem

Generalized Multi-Assignment Problem introduced by Park et al [54] is composed of tasks that can be assigned to more than one agent.

1.6 Conclusion

In this chapter, we have introduced general information of the Generalization Assignment Problem. It is known as an NP-complete problem and also is a large scale problem. Because it is a larger problem, there are many problems that are considered as particular

cases of Generalization Assignment Problem (GAP). The formulation of the classical Assignment Problem (AP) appears as a particular problem of the Generalization Assignment Problem (GAP) when n, number of tasks is equal to m, number of agents. The Generalized Assignment Problem (GAP) is known as generalization form of the Multiple Knapsack Problem (MKP) where the weight and profit of the items are independent of the knapsack. In chapter 3, we are giving more details for the Multiple Knapsack Problem (MKP).

Chapter 2

2 Knapsack Problem

2.1 Introduction

The 0 - 1 knapsack problem (KP) is known as an NP combinatorial optimization problem. It is considered as the simplest linear programming problem and appears in many applications in industry and financial management [43]. Furthermore, many algorithms such as Dynamic Programing, Branch and Bound algorithm and Genetic algorithm have been used to solve the $0 - 1$ knapsack problem. Actually, there are four main classes of algorithms solving the $0 - 1$ Knapsack Problem. Indeed, Bellman (1950) introduced the first algorithm using dynamic programming which improved the complexity to $O(nW)$. In addition, there are many variants of knapsack problem have been solved by using dynamic programming [43]. However, the capacity W can be an exponential function of n.

The second class of methods uses the Branch and Bound Algorithm (B&B) which is firstly introduced by Kolesar (1967) [32]. Between 1970 and 1979 many types of branch and bound algorithms were developed in order to solve KP with a high number of variables [43]. The most well-known approach of this period is due to Horowitz and Sahni [12]. Sahni [58] extended the result of Johnson (1974) and introduced the first polynomial time algorithm for the $0 - 1$ Knapsack Problem [43]. In 1980 Balas and Zemel proposed a new algorithm to solve the KP by sorting a small subset of the variables [43].

The algorithms of these first two classes are all exact methods while the last two are heuristics. The third class is constituted of algorithms which provide near optimal solution. The most popular is the Greedy Algorithm introduced by Dantzig (1957). The remaining class describes the evolutionary algorithm and particularly the Genetic Algorithm [69] which behave very well applied to some types of knapsack problem.

2.2 Mathematical Formulation of $0-1$ KP

Given a set of n items $j = 1, \dots, n$, each having a weight w_j and inducing a unit profit P_j, the knapsack problem consist in selecting some items to load a knapsack with a total capacity of W in order to maximize the total profit. The most common problem is called the $(0-1)$ Knapsack because it selects at most one of each type of item.

It can be formulated as

$$KP \begin{cases} \max Z = \sum_{j=1}^{n} P_j X_j \\ \sum_{j=1}^{n} w_j X_j \leq W \\ X_j \in \{0,1\}; \quad j = 1, \dots, n \end{cases}$$

Where X_j is a decision variable satisfying

$$X_j = \begin{cases} 1 & \text{if item } j \text{ is assigned to the knapsack} \\ 0 & \text{else} \end{cases} \quad j = 1, \dots, n$$

Also, we consider the conditions for the KP

P_j, w_j and W are non-negative integers

$$\sum_{j=1}^{n} w_j > W$$

$$w_j \leq W; \quad j = 1, \ldots, n$$

By using the cost instead of the profit a minimization formulation can be used.

2.3 Concept of Branch and Bound Algorithm

The branch and bound algorithm is considered as one of many algorithms that find an optimal solution for an integer programming problem. The main concept of the branch and bound approach is to partition the solution set into subsets solutions and select one of them having the highest value of the objective function. In addition, it consists of two most important aspects: branching and bounding.

To have a better understanding about how the branching and bounding procedures work, Kellerer et al [29] have described each procedure separately. First, they suppose a function and want to solve it as a maximization problem

$$\max_{y \in Y} f(y)$$

In the branching procedure, suppose $\widehat{Y}$ is the subset of the solutions set Y, and the algorithm partitions to some smaller subsets such as $Y_1, .., Y_n$. Since the union of all these subsets is given by $\widehat{Y}$. In this procedure, it will repeat the partition till finding only one feasible solution for each subset, and the best solution will be selected based on the present objective value. Using the bound process will give us the upper and lower bounds.

The Branch and Bound algorithm has been represented as a tree, whose nodes refer to the subsets solutions. In the branch process, the algorithm begins with a node that has not been selected, called a terminal node. The terminal node should have the highest value of the upper bound to be able to create two new nodes. The subset solution is represented as a node having an item, and this item divides to three types: included, no included and unassigned items. The included item means the solution of the set item is included in the node and otherwise non-included. Third is called unassigned and means the solution does not belong to the previous types [32].

After the branch operation starts with a node, the unassigned items will be selected; one node becomes included and another becomes non-included. The algorithm in this operation will work to check the feasibility of the solutions contained in the two new nodes. When the subset of the solutions contained in a given node is feasible; therefore, the upper bound will be obtained and the process continues, otherwise further branching is conducted from when the node is eliminated [32]. An optimal solution can be found when all items are sorted based on a decreasing order of the ratio P_j/w_j and then the algorithm starts by the first item and continues until the capacity W is reached [43].

2.4 The Branch and Bound Algorithm

To find a good solution for the knapsack problem; it requires to consider ratio of the profit to the weight ratio e_j of each item which is named the efficiency of an item with:

$$e_j = \frac{P_j}{w_j}$$

Since we suppose the items to be sorted by their efficiency in decreasing order such that

$$\frac{P_1}{w_1} \geq \frac{P_2}{w_2} \geq \cdots \geq \frac{P_n}{w_n}$$

In addition, we should know the place to do the branch, which is represented by the point where the branch technic occurs. We denote the branch point by s and is called split item. Where s is

$$\sum_{j=1}^{s} w_j X_j > W$$

Next, finding the optimal solution of linear programming relaxation. Therefore, the relaxation of knapsack problem obtained from "the Knapsack Problem formulation" by removing the integrality constraint on X_j [43].

$$KP \begin{cases} \max Z = \sum_{j=1}^{n} P_j X_j \\ \sum_{j=1}^{n} w_j X_j \leq W \\ 0 \leq X_j \leq 1; \quad j = 1, \ldots, n \end{cases}$$

Its solution provides an upper bound z_{ub}; the optimal solution of the relaxation of knapsack problem is

$$X_j = 1, \qquad j = 1, \ldots, s-1,$$

$$X_s = \frac{W - \sum_{j=1}^{s-1} w_j X_j}{w_s}$$

$$X_j = 0, \qquad j = s+1, \ldots, n$$

The upper bound of knapsack problem is

$$z_{ub} = \sum_{j=1}^{s-1} P_j + P_s \frac{W - \sum_{j=1}^{s-1} w_j X_j}{w_s}$$

In addition, to solve the knapsack problem by branch and bound algorithm; it needs to find a lower bound z_{lb}. Therefore, the lower bound can be obtained by using the Greedy heuristic. Where

$$X_j = 1; \qquad j = 1, \dots, (s-1)$$

and the set a value z_{lb} satisfying

$$z_{lb} \geq \sum_{j=1}^{s-1} P_j$$

In addition, the optimal solution z^* will be

$$\sum_{j=1}^{s-1} P_j \leq z_{lb} \leq z^* \leq z_{ub} \leq \sum_{j=1}^{s} P_j$$

After that, start with initial node X_j and for each node compute the upper bound if we can, the remaining capacity S_j and the solution $\bar{z}$. Therefore, check if node belongs $X_j = 1;\ j \in N_1$, $X_j = 0;\ j \in N_0$ and $F = \{1, \dots, n\} - N_0 - N_1$.

When $X_j = 1;\ j \in N_1$ that implies the node is included the solution otherwise $X_j = 0;\ j \in N_0$. Moreover, when the node does not belongs to the previous sets that mean the node is non-assigned. For each assigned node the remaining capacity S_j should be calculated by

$$S_j = W - \sum_{j=1}^{N_1} w_j$$

and $\bar{z}$ represents the solution of the node using the evaluation function

$$\bar{z} = \sum_{j=1}^{N_1} P_j$$

After we find all the above, we consider the restricted problem

$$\begin{cases} \max\limits_{X} Z = \sum\limits_{j=1}^{n} P_j X_j \\ \sum\limits_{j=1}^{n} w_j X_j \leq \bar{W} \\ X_j \geq 0; \quad j = 1, \dots, n \end{cases}$$

In addition, when we find the solution $\bar{z}$ for each node, we should compare between the solutions $\bar{z}$ and the upper of the restricted U_{ubr}. Therefore, if $\bar{z} \leq z_k$ this node is called probe that needs to perform backtracking. Since the backtracking consists of reversing the search by considering the last node X_k from which $X_k = 1$. After that, we consider the alternative by setting $X_k = 0$. The backtracking is done when we reach the last node of a branch or when the node is a probe and cannot be improved by going deeper.

In addition, when its weight $w_j > S_j$, the node will equal $x_j = 0$, and it becomes belong N_0 and not belongs F. However if $w_j < S_j$, the node will equal $x_j = 1$, and it becomes belong N_1, not belongs F, its capacity is $W \coloneqq W - w_j$ and its solution is $\bar{z} \coloneqq \bar{z} + P_j$.

Finally, when the solution of the node $\bar{z} > z_k$ then $z_{k+1} = \bar{z}$ becomes the solution for the node.

2.5 Steps of Branch and Bound Algorithm

1. Verify if the problem is nontrivially feasible by testing at least one index $j = 1,2, \dots, n$

$$w_j \leq W$$

When it is feasible, continue to next step otherwise stop.

2. Find the split item s or is called branch point by

$$\sum_{j=1}^{s} w_j \geq W$$

3. Calculate the upper bound of the problem using

$$z_{ub} = \sum_{j=1}^{s-1} P_j + P_s \frac{W - \sum_{j=1}^{s-1} w_j x_j}{w_s}$$

4. Compute the lower bound of the problem and that the solution should be satisfied the following

$$z_{lb} \geq \sum_{j=1}^{s-1} P_j$$

5. Start with initial node X_j and for each node compute the upper bound if we can, the remaining capacity S_j, and the solution z_j.

6. Behind initial node we decide next node and written if belongs N_1, N_0, or have not assigned yet i.e. belongs to F.

7. Perform backtracking if the node is probe until reach the last node of branch or when the node is probe and cannot be improved by going deeper.

2.6 Example of illustration

Consider the following problem with $n = 7$ items. The unit profit, weight and capacity are

$$P_j = (70, 20, 39, 35, 7, 5, 9)$$

$$w_j = (31, 10, 20, 18, 4, 3, 6)$$

$$W = 50$$

The knapsack problem is formulated as following:

$$\begin{cases} \max Z = 70X_1 + 20X_2 + 39X_3 + 35X_4 + 7X_5 + 5X_6 + 9X_7 \\ 31X_1 + 10X_2 + 20X_3 + 18X_4 + 4X_5 + 3X_6 + 6X_7 \leq 50 \\ X_j \in \{0,1\}; \quad j = 1, \dots, 7 \end{cases}$$

First, we need to find the spilt items s "branch point". Therefore, the value of s is $s = 3$ because $w_1 + w_2 + w_3 = 61 > 50$. Second, if we consider the solution problem, its optimal solution is

$$X_1 = 1,\ X_2 = 1, X_3 = \frac{50-31-10}{20} = \frac{9}{20} \text{ and } X_4 = X_5 = X_6 = X_7 = 0.$$

It is not feasible for the knapsack problem. However, it provides the upper bound

$$z_{ub} = 70 + 20 + 39\left(\frac{9}{20}\right) = 107.55$$

We can set the upper bound $U = 107$.

Third, we evaluate the lower bound for considering any feasible solution for example

$$X_1 = 1,\ X_2 = 1,\ X_3 = 0,\ X_4 = 0,\ X_5 = 1,\ X_6 = 1,\ X_7 = 0.$$

The lower bound is $L = 102$. We have the relation

$$L = 102 \leq z^* \leq 107 = U.$$

Now, for each node we have to find the upper and lower bound beginning with an initial node.

Iteration 1. We set $X_1 = 1$ because if we consider $X_1 = 0$ then the split item of the restrained problem is $s = 5$ and the upper bound becomes

$$z_{ub} = 20 + 39 + 35 + 7\left(\frac{50 - 48}{4}\right) = 97.5$$

We can set its upper bound $U_1 = 97$. Since the largest value $U_1 = 97 < 102$; therefore, the branch $x_1 = 0$ can be ignored. For that, the initial node is $X_1 = 1$ which has $z_1 = 70$ and $S_1 = 19$. Since the remaining capacity is $S_1 = 19$ then $X_3 = 0$ because $w_3 = 20 > S_1 = 19$. Therefore, the nodes are

$$N_1 = \{1\},\ N_0 = \{3\} \text{ and } F_1 = \{2, 4, 5, 6, 7\}.$$

The associated node N_1 is represented by $z_1 = 70$ and $S_1 = 19$.

Iteration 2. The smallest index of $F_1 = \{2, 4, 5, 6, 7\}$ is $i = 2$ since $w_2 = 10 < S_1 = 19$ we set $X_2 = 1$ and $S_2 = 9$. Then the node becomes

$$z_2 = 90, S_2 = 9 \quad \text{with} \quad N_1 = \{1, 2\},\ N_0 = \{3\} \quad \text{and} \quad F_1 = \{4, 5, 6, 7\}.$$

Iteration 3. The smallest index of $F_1 = \{4,5,6,7\}$ is $i = 4$ since $w_4 = 18 > S_2 = 9$ we set $X_4 = 0$. The node is represented by

$$z_2 = 90,\ S_2 = 9 \quad \text{with} \quad N_1 = \{1, 2\},\ N_0 = \{3,4\} \quad \text{and} \quad F_1 = \{\ 5, 6, 7\}.$$

Iteration 4. The smallest index of $F_1 = \{5,6,7\}$ is $i = 5$ since $w_5 = 4 < S_2 = 9$ we set $X_5 = 1$. The node is represented by

$$z_3 = 97, S_3 = 5 \quad \text{with} \quad N_1 = \{1, 2, 5\},\ N_0 = \{3,4\} \quad \text{and} \quad F_1 = \{6, 7\}.$$

Iteration 5. The smallest index of $F_1 = \{6,7\}$ is $i = 6$ since $w_6 = 3 < S_3 = 5$ we set $X_6 = 1$. The node is represented by

$$z_4 = 102 \quad \text{with} \quad S_4 = 2, N_1 = \{1, 2, 5,6\},\ N_0 = \{3,4\} \quad \text{and} \quad F_1 = \{7\}.$$

Iteration 6. The smallest index of $F_1 = \{7\}$ is $i = 7$ since $w_7 = 6 >\ S_4 = 2$ we set $X_7 = 0$. The node is represented by

$$z_4 = 102,\ S_4 = 2 \quad \text{with} \quad N_1 = \{1, 2, 5,6\}, N_0 = \{3,4,7\} \quad \text{and} \quad F_1 = \{\emptyset\}.$$

Iteration 7. Since $n = 7$ we backtrack by considering the last assigned variable. This provides

$X_6 = 1$, and we set its alternatives $X_6 = 0$. Therefore, the node becomes

$$z_3 = 97, S_3 = 5 \quad \text{with} \quad N_1 = \{1, 2, 5\},\ N_0 = \{3,4,6\} \quad \text{and} \quad F_1 = \{\ 7\}.$$

Iteration 8. The smallest index of $F_1 = \{7\}$ is $i = 7$ since $w_7 = 6 > S_3 = 5$; therefore, we set $X_7 = 0$, and the node represented by $z_3 = 97$ and $S_3 = 5$.

Iteration 9. Since $n = 7$ we backtrack by considering the last assigned variable. This provides $X_5 = 1$ and then we set its alternatives $X_5 = 0$. Therefore, the node becomes

$$z_9 = 90, S_9 = 9 \text{ with } N_1 = \{1,2\},\ N_0 = \{3,4,5\} \text{ and } F_1 = \{6,7\}.$$

Iteration 10. The smallest index of $F_1 = \{6,7\}$ is $i = 6$ since $w_6 = 3 < S_9 = 9$; therefore, we set $X_6 = 1$, and the node becomes

$$z_{10} = 95,\ S_{10} = 6 \text{ with } N_1 = \{1,2,6\},\ N_0 = \{3,4,5\} \text{ and } F_1 = \{7\}.$$

Iteration 11. The smallest index of $F_1 = \{7\}$ is $i = 7$ since $w_7 = 6 \leq S_{10} = 6$; therefore, we set $X_7 = 1$ and then the node becomes

$$z_{11} = 104, S_{11} = 0 \text{ with } N_1 = \{1,2,6,7\},\ N_0 = \{3,4,5\} \text{ and } F_1 = \{\emptyset\}.$$

Iteration 12. Since $n = 7$ we backtrack by considering the last assigned variable. This provides $x_7 = 1$ then we set its alternative $X_7 = 0$. Therefore, the node becomes

$$z_{12} = 95, S_{12} = 6 \text{ with } N_1 = \{1,2,6\},\ N_0 = \{3,4,5,7\} \text{ and } F_1 = \{\emptyset\}.$$

Iteration 13. Since $n = 7$ we backtrack to $X_6 = 1$ then we set $X_6 = 0$. This implies the node is $z_{21} = 90, S_{13} = 9$ and becomes probe $U = 99 < L < 102$.

Iteration 14. Since $n = 7$ we backtrack by considering the last assigned variable. This provides $X_2 = 1$ then we set its alternative $X_2 = 0$ and the node becomes

$$z_{14} = 95, S_{14} = 6 \quad \text{with} \quad N_1 = \{1\},\ N_0 = \{2,3\} \quad \text{and} \quad F_1 = \{4,5,6,7\}.$$

Iteration 15. The smallest index of $F_1 = \{4,5,6,7\}$ is $i = 4$ since $w_4 = 4 \leq S_{15} = 6$; therefore, we set $w_4 = 1$, and the node becomes

$$z_{15} = 105, S_{15} = 1 \quad \text{with} \quad N_0 = \{1,4\}, N_1 = \{2,3\} \quad \text{and} \quad F_1 = \{5,6,7\}.$$

Iteration 16, 17 and 18. Since $w_j = S_{15} = 1$ and $j = 5,6,7$ then iteration $16, 17,$ and 18 imply $X_5 = X_6 = X_7 = 0$.

Iteration 19. Since $n = 7$ we backtrack by considering the last assigned variable. This provides $X_4 = 1$ then we set its alternative $X_4 = 0$ and $S_{19} = 19$. The node becomes $z_{19} = 70$, $S_{19} = 19$ and becomes probe $U = 91 < L = 102$.

Iteration 20. We backtrack by considering the last assigned variable. This provides $X_1 = 1$ then the optimal solution is $z^* = 105$ where

$$X_1 = X_4 = 1;\ X_2 = X_3 = X_5 = X_6 = X_7 = 0.$$

Therefore, that achieves

$$L = 102 \leq 105 \leq 107 = U$$

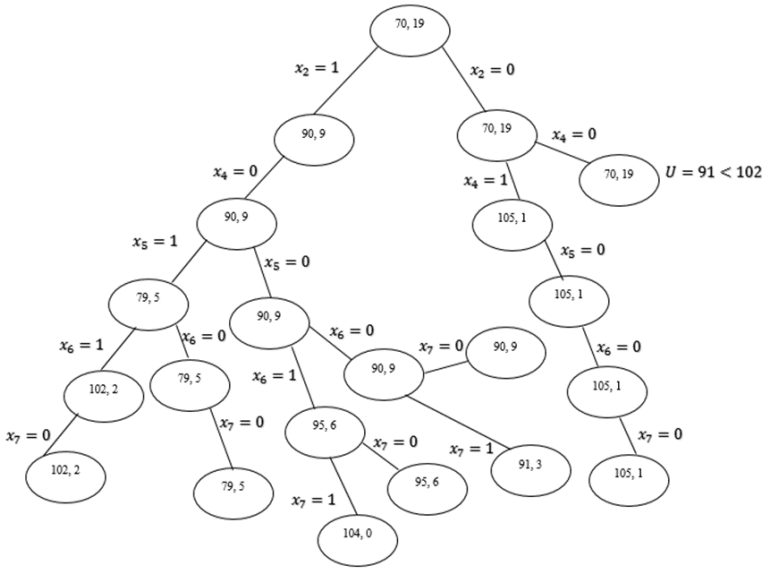

Figure 1: Tree Representation

2.7 Conclusion

In this chapter, we have presented the $0 - 1$ Knapsack Problem, which is a combinatorial optimization problem. In addition, we introduced Branch and Bound algorithm as a method to obtain the optimal solution for the problem. This method is also applied to an illustration example. However, there are also two main types of approaches developed to solve the Knapsack Problem. First, a Genetic Algorithm is considered as a suitable algorithm for solving the problem because Genetic Algorithm optimizes the huge number of solution that is available for solving the Knapsack Problem. For more

information, please refer to Najadat, F. A. [48]. Dynamic Programming Algorithm is also considered as an efficient method to solve the problem [45] [43].

Chapter 3

3 The $0-1$ Multiple Knapsack Problem

3.1 Introduction

The $0-1$ Multiple Knapsack Problem (MKP) is known as a generalization of the $0-1$ Knapsack Problem (KP) by considering more than one knapsack. It is also known as an NP-complete implying that MKP cannot be solved by polynomial time algorithm. It selects among n items to load m knapsacks in order to maximize the resulting total profit. Obviously, for each knapsack, the total weight of the selected items should not exceed its capacity. This problem has been solved by many algorithms such as Branch and Bound (B&B) algorithm and Dynamic Programming (DP) approach.

Ingargiola and Korsh [27] suggested a branch and bound approach which used a reduction procedure based on dominance relationships between pairs of items. In 1987, Hung and Fisk [26] introduced an approach based on Branch and Bound with depth – first strategy as a journey. They computed the upper bound by Lagrangian relaxation, with a reducing scheduling capacity W_i [63]. They also developed the algorithm of Martello and Toth [43]; therefore, they computed the upper bound by surrogate relaxation and taking the minimum of the lagrangian upper bounds and surrogate relaxation method [63]. Martello and Toth (1981) [42] developed their algorithm by proposing the Bound and Bound Algorithm (B&B) to solve the Multiple Knapsack Problem (MKP). They also called this algorithm Martello and Toth Method (MTM) and used to solve the problem with Greedy heuristics, which involves solving a series of problems with m single

knapsack. Pisinger [57] proposed a new algorithm which is called Mulknap and based on the algorithm (MTM). Funkunaga and Korf [18] proposed a new method called the bin – completion method which is based on the Branch and Bound algorithm.

For Dynamic programming, Martello and Toth [43] clarify this approach is impractical to solve the Multiple Knapsack problem (MKP).

In this chapter, we present the $0-1$ Multiple Knapsack Problem and focus on the Branch and Bound Algorithm (B&B) classes of methods.

3.2 The Formulation of the Multiple Knapsack Problem

The $0-1$ Multiple Knapsack Problem is defined by selecting among n items to load m knapsacks in order to maximize the resulting total profit, while for each knapsack, the sum of the weight w_j of the selected items should not exceed its capacity W_i.

The Mathematical formulation is as following:

$$MKP\begin{cases} \max Z = \sum_{i=1}^{m}\sum_{j=1}^{n} P_j X_{i,j} \\ \sum_{j=1}^{n} w_j X_{i,j} \le W_i \; ; \;\; i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} \le 1 \; ; \;\; j = 1, \dots, n \\ X_{i,j} \in \{0,1\} \; ; \;\; i = 1, \dots, m \; ; \;\; j = 1, \dots, n \end{cases}$$

The decision variables X_{ij} are such that $X_{i,j} = 1$ when item j is assigned to knapsack i and $X_{ij} = 0$ otherwise. The first equation says the total profit of assigning the item to

knapsack should be maximized. Second equation presents the total weight of item j should not exceed the knapsack's capacity. The last two equations show each item assigns to only one knapsack or not. In addition, we can set without loss of generality the following conditions for the MKP

$$P_j > 0\ ,\ w_j > 0 \text{ and } W_i > 0 \text{ are integers } \ j = 1, \dots, n; i = 1, \dots, m$$

$$w_j \leq \max\ \{W_i\ ;\ i = 1, \dots, m\}\ ;\ j = 1, \dots, n$$

$$W_i \geq \min\ \{w_j\ ; j = 1, \dots, n\};\ i = 1, \dots, m$$

$$\sum_{j=1}^{n} w_j > W_i\ ;\ \ i = 1, \dots, m$$

Furthermore, when $m = 1$ that give us the 0 -1 knapsack problem (KP) which is introduced in chapter 2. In addition, when each item j belongs profit P_j (instead of profit $P_{i,j}$), the problem becomes a particular case of the Generalized Assignment Problem where the weight of items are independent to the knapsack i.e. $w_{i,j} = w_j\ ; j = 1, \dots, n$ and $P_{i,j} = P_j\ ; j = 1, \dots, n.$

3.3 Relaxations of the $(0 - 1)$ MKP

There are different techniques to compute upper bounds for the Multiple Knapsack Problem: the Surrogate relaxation, the Lagrangian relaxation, and the linear programming relaxation.

3.3.1 Surrogate Relaxation

The Surrogate relaxation for the Multiple Knapsack Problem S(MKP, π) is defined to have a nonnegative vector $(\pi_i, \dots, \pi_m)$ of multipliers, so the formulation becomes:

$$S(MKP,\pi)\begin{cases} \max Z = \sum_{i=1}^{m}\sum_{j=1}^{n} P_j\, X_{i,j} \\ \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} w_j\, X_{i,j} \le \sum_{i=1}^{m} \pi_i\, W_i \\ \sum_{i=1}^{m} X_{i,j} \le 1;\ \ j = 1, \dots, n \\ X_{i,j} \in \{0,1\};\ \ i = 1, \dots, m\,; j = 1, \dots, n \end{cases}$$

The best choice of multipliers π_i is based on those that produce the minimum value of S(MKP, π) [57].

Martello and Toth [43] have provided a proof for any instance of Multiple knapsack Problem (MKP), the optimal choice of multipliers π_i for $i = 1, \dots, m$ is $\pi_i = k$, where k is a nonnegative constant. Therefore, the S(MKP, π) becomes as a single Knapsack problem and its formula as following

$$\begin{cases} \max Z = \sum_{j=1}^{n} P_j X_j \\ \sum_{j=1}^{n} w_j X_j \le W \\ X_j \ge 0; \quad j = 1, \dots, n \end{cases}$$

The capacity W represents the sum of all the knapsacks' capacities

$$W = \sum_{i=1}^{m} W_i$$

3.3.2 Linear Programming Relaxation

The linear programming relaxation is used to compute the upper bound of Multiple Knapsack Problem. The upper bound is obtained by relaxation the constraint

$$X_{ij,} \in \{0, 1\}\,;\ i = 1, \dots, m,\ \ j = 1, \dots, n$$

$$0 \leq X_{i,j} \leq 1\,;\ \ i = 1, \dots, m,\ \ j = 1, \dots, n$$

In addition, Martello and Toth [43] have provided a proof that an optimal solution for the Multiple Knapsack Problem using linear programming relaxation is equal to an optimal solution to the linear relaxed SMKP. Therefore, the upper bound of the Multiple Knapsack problem can be found using Dantzing's bound of the corresponding single Knapsack Problem [12] [57].

Let the items $j = 1, \dots, n$ are stored according to decreasing profit to weight rations

$$\frac{P_1}{w_1} \geq \frac{P_2}{w_2} \geq \cdots \geq \frac{P_n}{w_n}$$

And s represents as a split point and is defined by

$$s = \min\{\, j\colon \sum_{i=1}^{j} w_i \geq W \,\}$$

where the capacity W is equal

$$W = \sum_{i=1}^{m} W_i$$

Therefore, the Dantzig upper bound [12] [57] is provided

$$U_{MKP} = \sum_{j=1}^{s-1} P_j + \left[\left(\mathrm{W} - \sum_{j=1}^{s-1} w_j \right) P_s / w_s \right]$$

3.3.3 Lagangian Relaxation

The Lagrangian relaxation for the Multiple Knapsack Problem L(MKP, λ) is defined using a nonnegative vector $(\lambda_1, \dots, \lambda_n)$ of multipliers [43], so the formulation of L_1 (MKP, λ) becomes:

$$\begin{cases} \max Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_j X_{i,j} - \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{m} X_{i,j} - 1 \right) \\ \sum_{j=1}^{n} w_j X_{i,j} \le W_i; \quad i = 1, \dots, m \\ X_{i,j} \in \{0, 1\}; \;\; i = 1, \dots, m \, , j = 1, \dots, n \end{cases}$$

Also, can be written as

$$\max Z = \sum_{i=1}^{m} \sum_{j=1}^{n} \bar{P}_j X_{i,j} + \sum_{j=1}^{n} \lambda_j$$

Where

$$\bar{P}_j = P_j - \lambda_j; \quad j = 1, \dots, n$$

Therefore, Martello and Toth [43] proved that the Lagrangian relaxation might be decomposed into m independent $0-1$ Knapsack Problem (KP). Its formulation is

$$\begin{cases} \max\ Z_i = \sum_{i=1}^{m}\sum_{j=1}^{n} \bar{P}_j X_{i,j} \\ \sum_{j=1}^{n} w_j\, X_{i,j} \le W_i\,; \quad i = 1, \dots, m \\ X_{i,j} \in \{0,1\};\ \ i = 1, \dots, m\,, j = 1, \dots, n \end{cases}$$

All the Lagrangian relaxation problems have the same profits and weights, so the difference between these problems is just the capacity. And the optimal solution of Lagrangian relaxation Multiple Knapsack Problem $z(L_1(MKP, \lambda))$ becomes

$$z\big(L_1(MKP, \lambda)\big) = \sum_{i=1}^{m} z_i + \sum_{j=1}^{n} \lambda_j$$

In addition, Martello and Toth [43] showed there is no optimal choice of the multipliers λ, and an approximation of the optimal λ can be obtained by subgradient optimization technique. However, using this technique make the bounds computationally expensive to drive [57].

Hung and Fisk [24] used the complementary slackness conditions for λ_j given by

$$\bar{\lambda}_j = \begin{cases} P_j - w_j \dfrac{P_s}{w_s} & \text{if} \quad j < s; \\ 0 & \text{if } j \ge s \end{cases}$$

where s is the split point of $S(MKP)$ and is defined in section 3.4.2. With the choice λ_j, we have

$$\begin{cases} \bar{P}_j / w_j = P_s / w_s & \text{if } j \le s \\ \bar{P}_j / w_j = P_s / w_s & \text{if } j > s \end{cases}$$

And then

$$z\left(C\left(L_1\left(MKP, \bar{\lambda}\right)\right)\right) = \frac{P_s}{w_s}\sum_{i=1}^{m} W_i + \sum_{j=1}^{n} \bar{\lambda}_j$$

So we get

$$z\left(C\left(L_1\left(MKP, \bar{\lambda}\right)\right)\right) = z\left(C\left(S(MKP, 1)\right)\right) = z(C(MKP))$$

i.e. $\bar{\lambda}$ which indicates to the best multipliers for $C\left(L_1(MKP, \lambda)\right)$. Also, $L_1\left(MKP, \bar{\lambda}\right)$ and $S(MKP, 1)$ dominate the continuous relaxation. Therefore, there is no dominance between them [57].

Martello and Toth [43] show it is possible to find the second Lagrangian relaxation $L_2(MKP, \lambda)$.

With using a nonnegative vector $(\lambda_1, \dots, \lambda_n)$ of multipliers, so the formulation of $L_2(MKP, \lambda)$

$$\begin{cases} \max Z = \sum_{i=1}^{m}\sum_{j=1}^{n} P_j X_{i,j} - \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{m} w_j X_{i,j} - W_i\right) \\ \sum_{i=1}^{m} X_{i,j} \le 1; \quad j = 1, \dots, n \\ X_{ij,} \in \{0, 1\}; \quad i = 1, \dots, m\,, j = 1, \dots, n \end{cases}$$

However, $\lambda_j > 0$ that means not allow any multiplier to take the value zero. Because, if $\lambda_j = 0$, it produces a useless upper bound.

Also, can be written as

$$\max Z = \sum_{i=1}^{m} \sum_{j=1}^{n} (P_j - \lambda_j \, w_j) \, X_{i,j} - \sum_{i=1}^{m} \lambda_j W_i$$

Therefore, that presents the optimal solution can be obtained by selecting the knapsack $\hat{\imath}$ with a minimum value of λ_j and all items be chosen with $P_j - \lambda_{\hat{\imath}} \, w_j > 0$ for the knapsack $\hat{\imath}$. Since this is also the optimal solution of $C(L(MKP, \lambda))$, i.e.

$$z(C(L_2(MKP, \lambda))) = z(L_2(MKP, \lambda))$$

we have

$$z(L_2(MKP, \lambda)) \geq z(C(MKP))$$

and so this Lagrangian relaxation cannot produce a bound tighter than the continuous one.

However, Martello and Toth [43] showed the most natural polynomially – computable upper bound for the Multiple Knapsack Problem (MKP) is

$$U_1 = [z(C(MKP))] = [z(C(S(MKP)))] - \left[z(C\left(L\left(MKP, \bar{\lambda}\right)\right))\right]$$

Also, they proved the worst-case performance of bound U_1 is $m + 1$, i.e.

$$z(C(MKP)) \leq (m + 1) \, z(MKP)$$

3.4 Branch and Bound Algorithm

A depth- first Branch and Bound algorithm has proposed by Hung and Fisk (1978) [26] where using the Lgrangian Relaxation to obtain upper bounds, and branching was performed for the items which in the relaxed problem had been selected in most knapsacks.

At branching operation, each item assigned to the knapsacks in increasing index order, and the knapsacks were ordering in decreasing order

$$W_1 \geq W_2 \geq \cdots \geq W_m.$$

When all the knapsacks have been considered, the remained item was excluded from the problem.

Martello and Toth (1980) [41] proposed a different Branch and Bound Algorithm, where at each node, Multiple Knapsack Problem (MKP) was solved with constraint

$$\sum_{i=1}^{m} X_{i,j} \leq 1; \quad j = 1, \dots, n$$

dropped out, and the branching item was selected as an item which had been packed in $\hat{m} > 1$ knapsacks. $\hat{m}$ nodes are generated during the branching process by assigning the item to one of the corresponding $\hat{m} - 1$ knapsacks and by excluding it from these [43].

After that Martello and Toth have developed their algorithm, and they called it Bound and Bound Algorithm.

In the next section, we present the Bound and Bound Algorithm which is known as an adjustment of the Branch and Bound Algorithm.

3.5 Bound and Bound Algorithm

The Bound and Bound algorithm has proposed by Martello and Toth [43] [42] which is considered as a modification of the Branch and Bound method for Multiple Knapsack Problem (MKP). At each node, not only can be obtained an upper bound, but also a lower bound can be found.

In this algorithm the upper bounds are found by solving the surrogate-relaxed problem; however, the lower bounds are obtained by solving m individual knapsack problem as follows: the first knapsack $i = 1$ is filled optimally, the variables are selected will remove from the problem, and then the next knapsack $i = 2$ might be filled. This operation is repeated until all the knapsacks m have been filled [57].

The branching process follows this greedy solution as Martello and Toth showed that a greedy solution is better to guide the branching scheme than individual choices at each branching node.

Consequently, each node forks into two branching nodes, the first node assigning the next item j of a greedy solution to the chosen knapsack i, while the other branch excludes item j form knapsack i. An example below has solved by the Bound and Bound Algorithm.

3.5.1 Example of illustration

Consider the following problem with $n = 10$ items to load $m = 2$ knapsacks. Their profits, weights and capacities are

$$P_j = (78, 35, 89, 36, 94, 75, 74, 79, 80, 16);$$

$$w_j = (18, 9, 23, 20, 59, 61, 70, 75, 76, 30);$$

$$W_i = (103, 156)$$

This example is solved by applying the Bound and Bound Algorithm which is Branch and Bound method. First, to find the upper bound of the problem we are going to use the Surrogate relaxation. Since the bound and bound algorithm solve the Multiple Knapsack Problem to find the upper bound by the Surrogate relaxation as single Knapsack Problem. This implies

$$\begin{cases} \max Z = \sum_{j=1}^{10} P_j X_j \\ \sum_{j=1}^{10} w_j X_j \leq W \\ X_j \geq 0; \quad j = 1, \dots, n \end{cases}$$

where

$$W = \sum_{i=1}^{2} W_i = W_1 + W_2$$

$$= 103 + 156 = 256$$

The optimal solution of that surrogate problem is

$$X_j = (1, 0, 1, 1, 1, 1, 0, 0, 1, 0)$$

The upper bound is

$$z_{ub} = 78 + 89 + 36 + 94 + 75 + 80 = 452$$

We can set the upper bound $U = 452$

Second, we compute the lower bound by solving m individual knapsack problems

and a feasible solution is

$$(\bar{X}_{1,j}) = (1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0)$$

$$(\bar{X}_{2.j}) = (0, 1, 0, 0, 0, 1, 0, 0, 1, 0)$$

Therefore, the lower bound is $L = 451$

Now, finding the upper bound and lower bound for each node we have. However, before we find them, we need to begin with the initial node.

Iteration 1. We begin to set $X_{1,1} = 1$ for that, the initial node is $X_{1,1} = 1$ which has $z = 78$ and $W_1 = 85$ since the remaining capacity is $\bar{W} = 85$ then $X_{1,2} = 0$.

Iteration 2. The next node is $X_{1,3} = 1$ which has $z = 167$ and $W_1 = 62$ since the remaining capacity is $W_1 = 62$ then $X_{1,4} = 0$.

Iteration 3. The next node is $X_{1,5} = 1$ which has $z_3 = 261$ and $W_1 = 3$ since the remaining capacity is $W_1 = 3$ then $X_{1,6} = 0$.

Iteration 4. Since $n = 10$ we backtrack by considering the last assigned variable. This provides $X_{1,5} = 1$ then we set its alternative $X_{1,5} = 0$. Therefore, the node becomes $z = 167$ and $W_1 = 62$ and $U = 452$ and $L = 428$.

Iteration 5. The next node is $x_{1,2} = 1$which has $z = 202$ and $W_1 = 53$ and $U = 451$

Iteration 6. Since $n = 10$ we backtrack by considering the last assigned variable. This provides $X_{1,2} = 1$ then we set $x_{1,2} = 0$.

Therefore, the node becomes $z = 167$ and $W_1 = 62$ and $U = 452$ and $L = 452$.

Iteration 7. Since $n = 10$ we backtrack by considering the last assigned variable. This provides $X_{1,3} = 1$ then we set $X_{1,3} = 1$. Therefore, the node becomes $z = 78$ and $W_1 = 85$.

Iteration 8. Since $n = 10$ we backtrack by considering the last assigned variable. This provides $X_{1,1} = 1$ then we set $X_{1,1} = 0$. Therefore the node becomes $z = 103$ and $W_1 = 0$.

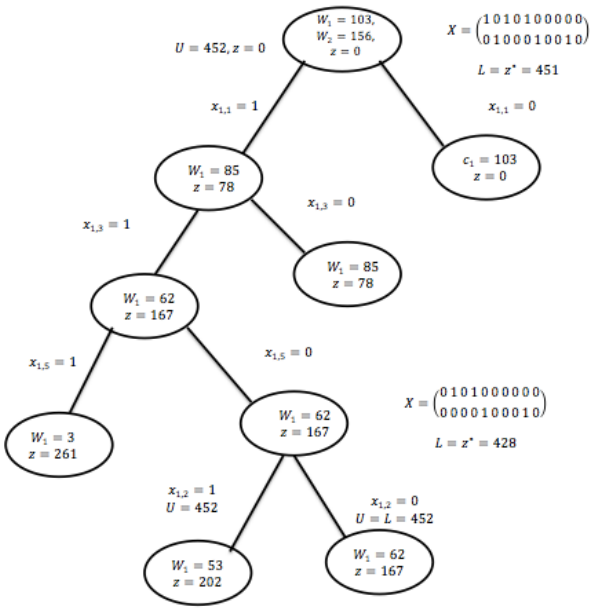

Figure 2: Tree representation [42]

3.6 Conclusion

In this chapter, we have presented the $0-1$ Multiple Knapsack Problem (MKP), which is known as a special case of the (GAP). The Branch and Bound Algorithmis presented in order to illustrate a method for solving the MKP. An example was then presented to illustrate the Bound and Bound method. In addition, there are other methods for solving the Multiple Knapsack Problem such as Genetic Algorithm and Dynamic Programming [66]. In the next two chapters, we are going to propose a new algorithm to solve the Knapsack Problem and Multiple Knapsack Problem.

Chapter 4

4 Solving the $0-1$ Knapsack Problem by an Adapted Transportation Algorithm

4.1 Introduction

In this chapter, we link the $0-1$ Knapsack Problem to the Linear Transportation Problem (LTP) then we solve the problem by using an adaptation of Transportation Algorithm (TA). The Vogel Approximation Method is applied to find an initial solution. It consists of assigning to each row and column a penalty which is the difference between the two least costs. The largest penalty indicates the line to be allocated first. Then the variable with least the cost on that line is assigned.

For the zero – one Knapsack Problem (KP) the Vogel Method is shown equivalent to the Greedy Algorithm. The initial solution is then improved by using the dual variable and resulting reduced cost. We prove that when no further reduction of the cost is possible, then we obtain an optimal solution.

4.2 Linear Transportation Problem

The transportation problem is considered as a special type of Linear Programming (LP), which is focused on studying the optimal transportation and allocation of resources [65]. The Linear Transportation Problem (LTP) is defined to ship a commodity from supply centers, called sources, to receiving centers, called destinations, while minimizing the

total distribution cost. Supposing that we have m sources $i = 1, \dots, m$ with W_i being the supply available to each source i and n destinations $j = 1, \dots, n$ with w_j being the demand for each destination j. P_{ij} is the cost of shipping one unit of commodity from source i to destination j. The transportation problem can be formulated as

$$LTP \begin{cases} \min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} \; X_{i,j} \\ \sum_{j=1}^{n} X_{i,j} = W_i; \quad i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} = w_j; \quad j = 1, \dots, n \\ X_{i,j} \geq 0; \quad i = 1, \dots, m; \; j = 1, \dots, n \end{cases}$$

It is a linear program with $m + n$ constraints and $m \times n$ variables and $X_{i,j}$ is a quantity moved from i to j.

The first equation represents the total cost of transporting a product from sources to destinations and should be minimized. The second one says the sum of all shipments that has shipping to a destination j should be equal to the supply. Equation three means the sum of all shipments that has shipping to a source i should be equal to the demand.

The Linear Transportation Problem is a balanced problem if the total supply equals the total demand.

$$\sum_{j=1}^{n} w_j = \sum_{i=1}^{m} W_i$$

Otherwise, it becomes unbalanced problem

$$\sum_{j=1}^{n} w_j \neq \sum_{i=1}^{m} W_i$$

4.3 Linear Transportation Problem and Knapsack Problem

The $0-1$ knapsack problem can be formulated as

$$KP \begin{cases} \max Z = \sum_{j=1}^{n} P_j X_j \\ \sum_{j=1}^{n} w_j X_j \leq W \\ X_j = \{0,1\}; \quad j = 1, \dots, n \end{cases}$$

Now, to link the 0-1 Knapsack problem to Linear Transportation Problem, there are two steps should follow.

First, the changing of variable $Y_j = w_j X_j$ implies a transportation problem representing a new formulation of Knapsack Problem

$$\begin{cases} \max Z = \sum_{j=1}^{n} \frac{P_j}{w_j} Y_j \\ \sum_{j=1}^{n} Y_j \leq W \\ Y_j \in \{0, w_j\}; \quad i = 1; \ j = 1, \dots, n \end{cases}$$

$$Y_j = \begin{cases} w_j & \text{when item } j \text{ assigned to knapsack } i \\ 0 & \text{else} \end{cases} \qquad j = 1, \dots, n$$

Second, the Knapsack Problem (KP) is not a balanced problem; therefore, we are going to add a dummy knapsack 2 associated to the vector

$$(Y_{2,1};\ Y_{2,2};\ \dots;\ Y_{2,n})$$

Since we have 2 knapsacks, we also consider the vector of variables

$$(Y_{1,1};\ Y_{1,2};\ \dots;\ Y_{1,n})$$

associated the knapsack by redefining

$$Y_j = Y_{1,j}$$

We also add a surplus item $n + 1$ associated to

$$(Y_{1,(n+1),};\ Y_{2,(n+1)})$$

Their coefficients in the maximization objective are equal to a zero value,

$$K_{2,j} = 0\ ;\ j = 1, \dots, n$$

which make the associated variables less attractive. Since the Knapsack Problem (KP) is not balanced, we add the weight of the surplus item $n + 1$ with zero coefficients and its weight equal to an unknown buffer B which is the non-allocated knapsack capacity i.e.

$$w_{(n+1)} = B$$

The total capacities of the knapsack and the dummy knapsack 2 are equal

$$W_1 = W \qquad \text{and} \qquad W_2 = B + \sum_{j=1}^{N} w_j - \sum_{i=1}^{M} W$$

The formulation of the balanced Knapsack Problem (BKP) becomes

$$BKP \begin{cases} \max Z = \sum_{j=1}^{n} \frac{P_j}{w_j} Y_{1,j} \\ \sum_{j=1}^{n+1} Y_{1,j} = W_1 \quad \text{and} \quad \sum_{j=1}^{n+1} Y_{2,j} = W_2 \\ \sum_{i=1}^{2} Y_{i,j} = w_j \,; \quad j = 1, \dots, (n+1) \\ Y_{i,j} \in \{0, w_j\}; \quad i = 1, 2 \,; \; j = 1, \dots, (n+1) \end{cases}$$

We consider the largest efficiency rate

$$K = \max_j \left\{ \frac{P_j}{w_j} \right\}$$

Then we subtract all the coefficients from that largest value to obtain new coefficients

$$K_{1,j} = K - \frac{P_j}{w_j} \quad \text{and} \quad K_{2,j} = K; \;\; j = 1, \dots, n$$

This transforms the Knapsack Problem into a minimization transportation problem and gets the new coefficient

$$MIN-KP\begin{cases} \min Z_L = \sum_{i=1}^{2}\sum_{j=1}^{n+1} K_{i,j}\, Y_{i,j} \\ \sum_{j=1}^{n+1} Y_{1,j} = W_1 \quad \text{and} \quad \sum_{j=1}^{n+1} Y_{2,j} = W_2 \\ \sum_{i=1}^{2} Y_{i,j} = w_j\,; \quad j = 1, \dots, (n+1) \\ Y_{i,j} \in \{0, w_j\}; \quad i = 1, 2\,;\ j = 1, \dots, (n+1) \end{cases}$$

It can be solved by an adaptation of the Transportation Algorithm (TA) presented in the next section.

4.4 Adapted Transportation Algorithm

4.4.1 Vogel Approximation Method

Vogel Approximation Method (VAM) is one of the bests heuristic method to find a initial basic solution for the Transportation Problem. In 1958, Vogel Approximation Method developed by William R. Vogel. It depends on the concept of computing penalty cost, which is defined by finding the difference between the two minimum costs for each row and column.

The steps below are given more explanations:

1. Compute the penalty cost for each row and column; which is given by the difference between two minimums cost.
2. Determine the largest penalty of the line to be assigned.
3. Assign the variable having the lowest unit cost.

4. Update the supply and demand.

   If there is one line remaining fill it and end

   Else continue

5. Cross out the assigned column or row with zero supply or demand and return to step 1.

Remark 1. If the items are ordered such that the efficiency rate is decreasing, the variable to be assigned is the first variable of the first column of the remaining table.

Remark 2. If the items are ordered such that the efficiency rate is decreasing, the initial solution is equivalent to the greedy approximation solution (Dantzig, 1957).

4.4.2 Dual Variable and test of reduction

The dual variables u_i and v_j associated to the current solution are provided by the following system of equations optimal

$$K_{i,j} - u_i - v_j = 0 \qquad \forall\, Y_{i,j} \in B$$

Since there are $m + n$ unknown variables and $m + n - 1$ equations, by setting $u_1 = 0$ then we determine a solution of dual variables and then the reduced cost of all non-basic variables

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \;\; ; \quad \forall\, Y_{i,j} \notin B$$

where B is the set of basic variables

We can notice that the current solution is optimal if the reduced cost for all non-basic variables is positive i.e.

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad \geq 0.$$

Otherwise there exist at least one non-basic variables $Y_{i,j}$ such that

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad < 0$$

In the case of the knapsack problem, the reduced cost of such variables satisfied

$$Y_{1,j} = 0, \;\; Y_{2,j} > 0 \quad \text{and} \quad \widehat{K}_{1,j} = K_{1,j} - K < 0$$

This indicates the possibility to decrease the total transportation cost. This will be tested by considering the move from $Y_{2,j}$ to $Y_{1,j}$ called the j-move. That move decreases the cost by

$$R_j = \widehat{K}_{1,j} \;\; Y_{2,j} \;\; < 0 \, .$$

Since the first constraint is limited by the Knapsack capacity W the j-move yields the use of the surplus and some minimal necessary moves of variables

$$Y_{1,\sigma(t)} \text{ to } Y_{2,\sigma(t)}; \; t = 1, \dots, T$$

where T is represent a move of column. This implies an increase of the cost equal to

$$I_j = \sum_{t=1}^{T} \widehat{K}_{2,\sigma(t)} \cdot Y_{1,\sigma(t)} > 0$$

The value $R_j + I_j$ represents the change of the cost function resulting in the j-move. Therefore if $R_j + I_j < 0$ then the objective cost is reduced and the move retained.

Otherwise the j-move is discarded. The comparison of these successful move progressively performed in order to retain the optimal one which is associated to the optimal solution.

4.4.3 Adapted Transportation Algorithm

In this section, we propose the adapted transportation algorithm

Step 0. **Ordering the items**

Order the items such that the efficiency rate $\frac{p_j}{w_j}$ is decreasing

Set $K = K_{1,1}$ be the maximal efficiency rate

Update the coefficients to change the KP into minimization

$$K_{i,j} := K - K_{i,j}; \quad i = 1,2 ; \quad j = 1, \dots, n$$

Step 1. **Initial solution and reduced costs**

Set $S_1 = 0$; $\overline{W} = W$ and s=1

For $j = 1, \dots, n$ do

If $w_j \leq \overline{W}$ then set $Y_{1,j} = w_j$,

$$S_{(j+1)} = S_j + w_j \text{ , } \overline{W} = W - S_{(j+1)} \text{ and } \widehat{K}_{2,j} = K - K_{1,j}$$

Else set $Y_{2,j} = w_j$, $\widehat{K}_{1,j} = K_{1,j} - K$,

$TC(s) = j$ and s:=s+1

EndIf

Endfor

Set $Y_{1,(n+1)} = \overline{W}$ and $Y_{2,(n+1)} = 0$

Step 2. **First decreasing evaluation**

Set $N = s - 1$

For $k = 1, \dots, N$ do

Set $j = TC(k)$ and $R_j = \widehat{K}_{1,j} \cdot X_{2,j} < 0$

Find $\sigma(t)$; $t = 1, \dots, T$ which is the minimal necessary move of variables $Y_{1,\sigma(t)}$ to $Y_{2,\sigma(t)}$; $t = 1, \dots, T$ resulting from the move of $Y_{2,j}$ to $Y_{1,j}$ and set

$$I_j = \sum_{t=1}^{T} \widehat{K}_{2,\sigma(t)} \cdot Y_{1,\sigma(t)} > 0$$

Evaluate

$$R_j + I_j = \widehat{K}_{1,j} \cdot Y_{2,j} + \sum_{t=1}^{T} \widehat{K}_{2,\sigma(t)} \cdot Y_{1,\sigma(t)}$$

If $R_j + I_j < 0$ then retain the move as a possibility to decrease the objective function

Set $k = N + 1$

Else discard the move

EndFor

Step 3. **Improving the initial solution**

For $l = (j + 1), \ldots, N$ do

Set $k = TC(l)$

If $Y_{2,j} < Y_{2,k}$ then do

Set $R_k = \widehat{K}_{1,k} \cdot Y_{2,k}$

If $\widehat{K}_{1,j} \cdot Y_{2,j} < \widehat{K}_{1,k} \cdot Y_{2,k}$ then the $k-$ move is discarded

Else

If $Y_{2,k} - Y_{2,j} \leq Y_{1,(n+1)}$ then accept the $k-$ move and set $\bar{k} = k$

Else

Find first r such that

$$Y_{2,t} \leq Y_{1,(n+1)} + Y_{2,t} + Y_{2,(t-1)} + \cdots + K_{2,r}$$

Set $I_k = \widehat{K}_{2,k-1} \cdot Y_{1,k-1} + \cdots + \widehat{K}_{2,r} \cdot Y_{1,r}$

If $I_k + R_k \leq I_j + R_j$ then discard the $k-$move

Else

Accept the $k-$move and set $\bar{k} = k$

EndIf

EndIf

EndIf

Else $(Y_{2,j} \geq Y_{2,k})$

Then $\widehat{K}_{1,j} \cdot Y_{2,j} < \widehat{K}_{1,k} \cdot Y_{2,k}$

If $Y_{2,j} - Y_{2,k} \leq Y_{1,(n+1)}$ then discard the $k-$move

Else

Find first r such that $Y_{2,r} - Y_{2,k} \leq Y_{1,(n+1)}$

Set $I_k = I_j - \widehat{K}_{2,k-1} \cdot Y_{1,k-1} - \cdots - \widehat{K}_{2,r} \cdot Y_{1,(r-1)}$

If $I_k + R_k < I_j + R_j$

then accept the $k-$move and set $\bar{k} = k$

Else

Discard the $k-$ move

EndIf

EndIf

EndIf

EndIf

Step 4. **Perform the optimal move**

The optimal move is associated to $\bar{k}$

Perform that optimal move which is from $Y_{2,\bar{k}}$ to $X_{1,\bar{k}}$

Here are some propositions:

Proposition 1. If $X_{1,k} = w_k > w_j$ then the move of $Y_{1,k}$ to $Y_{2,k}$ is not included in the improving j-move of the weight w_j from the variable $Y_{2,j}$ to $Y_{1,j}$.

Proof. Since $Y_{1,k} = w_k > w_j$ and $K_{1,j} > K_{1,k}$ then

$$\left|\widehat{K}_{1,j}\right| = K_{1,1} - K_{1,j} < \widehat{K}_{2,k} = K_{1,1} - K_{1,k} \quad \text{and} \quad \widehat{K}_{2,k} \cdot Y_{1,k} + \widehat{K}_{2,j} \cdot Y_{1,j} \geq 0$$

Therefore the move of $Y_{1,k}$ to $Y_{2,k}$ cannot be included in an improving and feasible j-move.

Proposition 2. Let's consider $Y_{1,k} = w_k$ and the move of w_k from $Y_{1,k}$ to $Y_{2,k}$ such that

$$R_j + I_j + (\widehat{K}_{2,k} \cdot Y_{1,k}) \geq 0.$$

Then that move cannot be included in the improving j-move.

Proof. If the move of $Y_{1,k}$ to $Y_{2,k}$is included then the j-move will increases the objective because $R_j + I_j + \left(\widehat{K}_{2,k} \cdot Y_{1,k}\right) \geq 0$ and will not improve it.

Proposition 3. Let's consider $Y_{1,k} = w_k$ such that

$$w_k + V_j \geq w_j \quad \text{and} \quad R_j + \bar{I}_j + \left(\widehat{K}_{2,k} \cdot Y_{1,k}\right) < 0.$$

Then the j-move is accepted. Moreover if the j-move is such that

$$w_k + V_j - w_j = 0$$

then the current solution yield by the j-move is optimal.

Proof. Since $w_k + V_j \geq w_j$ then the move of the basic variables associated to V_j provides a feasible j-move which yields an improved current solution. Therefore the j-move is accepted.

With the second condition, all variables $r > j$ are non-basic variables with negative reduced cost i.e.

$$Y_{1,r} = 0,\ Y_{2,r} = w_r \quad \text{and} \quad \widehat{K}_{1,r} = K_{1,r} - K_{1,1} < 0$$

Therefore, any r-move will involve the move of a total weight at least equal to w_r from basic variables $X_{1,k};\ k < j$ to $X_{2,k}$. Since the minimal reduced cost associated to these moves are higher to the absolute value of the negative reduced cost of any non-basic variables $r > j$ then the objective function can no longer be improved.

Now, we provide an example to illustrate the algorithm.

4.5 Example of illustration

We are going to use the same example that has been solved by the Branch and Bound in the previous chapter. The parameters of the Knapsack Problem are

$$n = 7, \qquad P_j = (70, 20, 39, 35, 7, 5, 9)$$

$$w_j = (31, 10, 20, 18, 4, 3, 6) \qquad W = 50$$

The mathematical formulation is

$$\begin{cases} \max Z = \sum_{j=1}^{7} P_j X_j \\ \sum_{j=1}^{7} w_j X_j \leq 50 \\ X_j = \{0, 1\}; \quad j = 1, \dots, 7 \end{cases}$$

It can be represented by the following table:

70	20	39	35	7	5	9	**50**
31	**10**	**20**	**18**	**4**	**3**	**6**	

Now, by the change of variable $Y_j = w_j X_j$ the formulation of the problem becomes

$$\begin{cases} \max Z = \sum_{j=1}^{7} \frac{P_j}{w_j} Y_{1,j} \\ \sum_{j=1}^{8} Y_{1,j} = 50 \\ \sum_{j=1}^{8} Y_{2,j} = B + 42 \\ \sum_{i=1}^{2} Y_{i,j} = w_j \; ; \;\; j = 1, \dots, 8 \\ Y_{i,j} \in \{0, w_j\}; \qquad i = 1,2 \; ; \;\; j = 1, \dots, 8 \end{cases}$$

The associated table representation is

$\frac{70}{31}$	$\frac{20}{10}$	$\frac{39}{20}$	$\frac{35}{18}$	$\frac{7}{4}$	$\frac{5}{3}$	$\frac{3}{2}$	**50**
31	**10**	**20**	**18**	**4**	**3**	**6**	

The problem is not balanced because

$$W = 50 \neq \sum_{j=1}^{7} w_j = 92$$

To balance the problem, we need to add a dummy knapsack 2 and a surplus item 8.

The total demand of item 8 is equal to a buffer B where B is unknown value.

$$w_8 = B$$

The total supply of knapsack 2 is equal

$$W_2 = B + \sum_{j=1}^{7} w_j - W = B + 92 - 50 = B + 42$$

Now the problem becomes balanced because

$$W_1 + W_2 = 50 + (B + 92 - 50) = B + 92 = \sum_{j=1}^{8} w_j$$

The formulation of BKP becomes

$$BKP \begin{cases} \max Z = \sum_{j=1}^{7} P_j \, Y_{1,j} \\ \sum_{j=1}^{8} Y_{1,j} = 50 \\ \sum_{j=1}^{8} Y_{2,j} = B + 42 \\ \sum_{i=1}^{2} Y_{i,j} = w_j \,; \;\; j = 1, \dots, 8 \\ Y_{i,j} \in \{0, w_j\}; \quad i = 1,2 \,; \;\; j = 1, \dots, 8 \end{cases}$$

The coefficient associated to the dummy knapsack and dummy item are equal zero. The table representation becomes

70	20	39	35	7	5	9	0	**50**
0	0	0	0	0	0	0	0	$\boldsymbol{B + 42}$
31	**10**	**20**	**18**	**4**	**3**	**6**	$\boldsymbol{B}$	

Now, changing the problem to minimization problem by considering the highest efficiency $\frac{70}{31}$. The formulation becomes the minimization and yields to the following transportation problem

$$\begin{cases} \min Z_L = \sum_{j=1}^{7} \left(\frac{70}{31} - \frac{P_j}{w_j} \right) Y_{1,j} \\ \sum_{j=1}^{8} Y_{1,j} = 50\,; \\ \sum_{j=1}^{8} Y_{2,j} = B + 42; \\ \sum_{i=1}^{2} Y_{i,j} = w_j \\ Y_{i,j} \in \{0, w_j\}; \quad i = 1,2\,;\ j = 1, \dots, 8 \end{cases}$$

This implies the following table

0	$\frac{8}{31}$	$\frac{191}{620}$	$\frac{175}{558}$	$\frac{63}{124}$	$\frac{55}{93}$	$\frac{47}{62}$	$\frac{70}{31}$	**50**
$\frac{70}{31}$	$\frac{70}{31}$	$\frac{70}{31}$	$\frac{70}{31}$	$\frac{70}{31}$	$\frac{70}{31}$	$\frac{70}{31}$	$\frac{70}{31}$	$\boldsymbol{B + 42}$
31	**10**	**20**	**18**	**4**	**3**	**6**	$\boldsymbol{B}$	

By using the approximation of the coefficients the table becomes

0	2.26	0.308	0.314	0.51	0.59	0.76	2.258	**50**
2.258	2.258	2.258	2.258	2.258	2.258	2.258	2.258	$\boldsymbol{B + 42}$

31	**10**	**20**	**18**	**4**	**3**	**6**	***B***	

To solve the transportation problem, we first find an initial solution by using the Vogel Approximation Method (VAM). It consists in assigning to each row and column a penalty which is the difference of the two lowest cost of that line.

This leads to

$$p_1 = 0.26 \; ; p_2 = 0; q_1 = 2.258 \; ; q_2 = 1.998 \; ; q_3 = 1.95 \; ; q_4 = 1.95 \; ; q_5 = 1.75 \; ;$$

$$q_6 = 1.67; q_7 = 1.5 \text{ and } q_8 = 0$$

Assignment 1. Since the largest penalty is, $q_1 = 2.258 - 0 = 2.258, j = 1, \dots, 8$; the variable to be assigned is

$$Y_{1,1} = 31 \qquad \text{and} \qquad S_1 = 50 - 31 = 19$$

Then column 1 is crossed out from the table.

Assignment 2. Since the largest penalty is associated to a column and the efficiency rate is decreasing; the next variable to be assigned is

$$Y_{1,2} = 10 \qquad \text{and} \qquad S_2 = 19 - 10 = 9$$

Then column 2 is crossed out from the table.

Since the remaining supply is lower than the weight these variables cannot be assigned. We set

$$Y_{2,3} = 20 \quad \text{and} \quad Y_{2,4} = 18.$$

Then we crossed out column 3 and 4 out from the table.

Assignment 3. Since the efficiency rate is decreasing, the next variable to be assigned is

$$Y_{1,5} = 4 \quad \text{and} \quad S_3 = 9 - 4 = 5$$

Then column 5 is crossed out from the table.

Assignment 4. Since the efficiency rate is decreasing, the next variable to be assigned is

$$Y_{1,6} = 3 \quad \text{and} \quad S_4 = 5 - 3 = 2$$

Then column 6 is crossed out from the table. Since $S_4 = 2 < w_7 = 6$ then the remaining variable Y_{17} cannot be assigned. We set $Y_{2,7} = 6$ and fill the remaining variable by setting

$$Y_{1,8} = 2;\ Y_{2,3} = 20;\ Y_{2,4} = 18;\ Y_{2,7} = 6;\ Y_{2,8} = 0$$

The initial feasible solution is presented in the following table with the total profit begin equal to

$$Z = 102 \quad \text{with} \quad \text{the buffer } B = 2$$

31	10			4	3		2
		20	18			6	0

To find the optimal solution, we need to find the dual variables $u_i \; and \; v_j$ for all basic variables.

We set $u_1 = 0$ and then we have

$$\begin{cases} u_1+v_1 = 0 & \Rightarrow\ v_1 = 0 \\ u_1 + v_2 = 0.26 & \Rightarrow\ v_2 = 0.26 \\ u_1 + v_5 = 0.51 & \Rightarrow\ v_5 = 0.51 \\ u_1 + v_6 = 0.59 & \Rightarrow v_6 = 0.59 \\ u_1 + v_8 = 2.258 & \Rightarrow v_8 = 2.258 \end{cases}$$

Since we know $v_8 = 2.258$; therefore, we find $u_2 = 0$. Also,

$$\begin{cases} v_3 = 2.258 \\ v_4 = 2.258 \\ v_7 = 2.258 \end{cases}$$

After that, we need to find the reduced cost of all non – basic variables

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad ; \quad \forall\, Y_{i,j} \notin B$$

The costs for the non – basic variables in the first row are

$$\widehat{K}_{1,3} = K_{1,3} - u_1 - v_3 \quad \Rightarrow \quad 0.308 - 2.258 = -1.95$$

$$\widehat{K}_{1,4} = K_{1,4} - u_1 - v_4 \quad \Rightarrow \quad 0.314 - 2.258 = -1.94$$

$$\widehat{K}_{1,7} = K_{1,7} - u_1 - v_7 \quad \Rightarrow\ 0.76 - 2.258 = -1.5$$

For the second row

$$\widehat{K}_{2,1} = K_{2,1} - u_2 - v_1 \quad \Rightarrow \quad 2.258 - 0 - 0 = 2.26$$

$$\widehat{K}_{2,2} = K_{2,2} - u_2 - v_2 \quad \Rightarrow \quad 2.258 - 0 - 0.26 = 2$$

$$\widehat{K}_{2,5} = K_{2,5} - u_2 - v_5 \quad \Rightarrow \quad 2.258 - 0 - 0.51 = 1.75$$

$$\widehat{K}_{2,6} = K_{2,6} - u_2 - v_6 \quad \Rightarrow 2.258 - 0 - 0.59 = 1.67$$

The current solution with reduced cost is presented below

31	**10**	−1.95	−1.94	**4**	**3**	−1.5	**2**
2.26	2	**20**	**18**	1.75	1.67	**6**	**0**

Since the most negative value is $\widehat{K}_{1,3} = -1.952$, then the move of $Y_{2,3} = 20$ to $Y_{1,3}$ will reduce the objective by

$$R_3 = 20 \times (-1.95) = -39$$

This will force at least the move of $Y_{1,1} = 31$ to $Y_{2,1}$ and the increasing

$$I_3 = 31 \times 2.26 = 70.02$$

Since adding $R_3 + I_3 = 31.02 > 0$ then the move of $Y_{2,3} = 20$ is not improving and is discarded by setting $C_{2,3} = 2.26 + 1 = 3.26$.

Since the next least reduced cost is $\widehat{K}_{1,4} = -1.94$, then the move of $Y_{2,4} = 18$ to $Y_{1,4}$ will reduce the objective by

$$R_4 = 18 \times (-1.94) = -35$$

This will force at least the move of $Y_{1,6} = 3$, $Y_{15} = 4$ and $Y_{1,2} = 10$. This implies an increase

$$I_4 = 10 \times (2) + 4 \times (1.75) + 3 \times (1.67) = 32$$

Since adding $R_4 + I_4 = -3 < 0$ then the move of $Y_{2,4} = 18$ is maintained.

The last negative reduced cost $\widehat{K}_{1,7} = -1.5$ yields the move of $Y_{2,7} = 6$ to $Y_{1,7}$ which reduces the objective by

$$R_7 = 6 \times (-1.5) = -9$$

This will force at least the move of $Y_{1,5} = 4$ and then the increasing

$$I_7 = 4 \times (1.75) = 7$$

Since adding $R_7 + I_7 = -2 < 0$ then the move of $Y_{2,7} = 6$ is maintained.

By comparing these two feasible solutions, the largest reduction of the cost function is provided by the move of $Y_{2,4} = 18$ to $Y_{1,4}$ which is the selected one. We notice that the move of any of the variables $Y_{2,6} = 6$ to $Y_{2,5} = 4$ and $Y_{2,2} = 10$ will imply the pervious situation which will not be an improvement and s discarded by setting $C_{2,6} = C_{2,5} = C_{2,2} = 3.26$. The move of $Y_{2,7} = 6$ to $Y_{1,7}$ is already tested and compared by the selected one. It will not provide improvement and is discarded by setting $C_{1,7} = 3.26$. Then the optimal solution with the total profit begin 105 is represented in the following table

31	+	+	**18**	+	+	+	**1**	**50**
2.26	**10**	**20**	1.946	**4**	**3**	**6**	**1**	$\boldsymbol{B + 42}$

4.6 Conclusion

In this chapter, we have presented the Knapsack Problem (KP) as a Linear Transportation Problem (LTP) and provide a new approach for solving $0-1$ Knapsack Problem (KP). To find the initial solution the Vogel Method (VAM) is used and shown to be equivalent to the Greedy Algorithm for the knapsack problem. Then an Adapted Transportation Algorithm (ATA) is applied to find an optimal solution. The approach can also be extended to some variants of the knapsack problem such as the Subset – sum problem ($P_j = w_j$) [7], the bound knapsack problem and the knapsack – like problems. It also can be generalized to solve the Zero – One Multiple Knapsack Problem (MKP) and the multiple Subset – sum problem. In next chapter, we will apply this method to solve the $0-1$ Multiple Knapsack Problem (MKP).

Chapter 5

5 Solving the $0-1$ Multiple Knapsack Problem by an Adapted Transportation Algorithm

5.1 Introduction

In this chapter, we link the $0-1$ Multiple Knapsack Problem to the Linear Transportation Problem (LTP). Then we solve the problem by using an Adaptation of Transportation Algorithm. The Vogel Approximation Method is applied to find an initial solution. It consists of assigning to each row and column a penalty which is the difference between the two minimum costs. The largest penalty indicates the line to be allocated first. Then the variable with minimum cost on that line is assigned. The initial solution is then improved by using the dual variable and resulting reduced cost. We prove that when no further reduction of the cost is possible, then we obtain an optimal solution.

5.2 The Multiple Knapsack Problem Formulation

It is defined by selecting among n items to load m knapsacks in order to maximize the resulting total profit while for each knapsack, the sum of the weight w_j of the selected items should not exceed its capacity W_i. The Multiple Knapsack Problem (MKP) is formulated as following:

$$MKP \begin{cases} \max \ Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_j X_{i,j} \\ \sum_{j=1}^{n} w_j X_{i,j} \leq W_i; \quad i = 1, \dots, m \\ \sum_{i=1}^{m} X_{ij,} \leq 1; \quad j = 1, \dots, n \\ X_{i,j} \in \{0,1\};\ i = 1, \dots, m\ ;\ \ j = 1, \dots, n \end{cases}$$

We consider the following conditions [43] without a loss of generality.

$$P_j > 0\ ,\ w_j > 0 \text{ and } W_i > 0 \text{ are integers } \ j = 1, \dots, n; i = 1, \dots, m$$

$$w_j \leq \max\ \{W_i\ ;\ i = 1, \dots, m\}\ ;\ j = 1, \dots, n$$

$$W_i \geq \min\ \{w_j\ ; j = 1, \dots, n\};\ i = 1, \dots, m$$

$$\sum_{j=1}^{n} w_j > W_i\ ;\ \ i = 1, \dots, m$$

5.3 Linear Transportation Problem and Multiple Knapsack Problem

In this section, we link the $0-1$ Multiple Knapsack problem to Linear Transportation Problem. It consists of presenting the MKP as Transportation Problem and having it

balanced. To present it as a Transportation Problem we change the decision variable $X_{i,j}$. The changing of variable $Y_{i,j} = w_j X_{i,j}$ implies

$$Y_{i,j} = \begin{cases} w_j & \text{when item } j \text{ is assigned to knapsack } i \\ 0 & \text{else} \end{cases} \qquad j = 1, \dots, n$$

Therefore, the new formulation of Multiple Knapsack – Transportation Problem is

$$MKP \begin{cases} \max \ \mathrm{Z} = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{P_j}{w_j} Y_{i,j} \\ \sum_{j=1}^{n} Y_{i,j} \le W_i; \qquad i = 1, \dots, m \\ \sum_{i=1}^{m} Y_{i,j} \le w_j; \qquad j = 1, \dots, n \\ Y_{i,j} \in \{0, w_j\}; \quad i = 1, \dots, m \ ; \ j = 1, \dots, n \end{cases}$$

Since the problem is not usually balanced, we add a dummy knapsack $(m+1)$ and associated to the variables

$$\left(Y_{(m+1),1};\ Y_{(m+1),2}; \dots;\ Y_{(m+1),n}\right)$$

Also, we add a dummy item $(n+1)$ associated to the variables

$$\left(Y_{1,(n+1)};\ Y_{2,(n+1)}; \dots;\ Y_{m,(n+1)}\right)$$

Their coefficients in the objective function are equal to zero value, which make this variable less attractive. The demand surplus of item $(n+1)$ is equal to an unknown buffer B

$$w_{(n+1)} = B$$

The supply for dummy knapsack $m+1$ is equal

$$W_{(m+1)} = B + \sum_{i=1}^{m} W_i - \sum_{j=1}^{n} w_j$$

The formulation of the balanced Multiple Knapsack Problem (BMKP) becomes

$$[\text{BMKP}] \begin{cases} \max \ \text{Z} = \sum_{i=1}^{m+1} \ \sum_{j=1}^{n+1} \frac{P_j}{w_j} \ Y_{i,j} \\ \sum_{j=1}^{n+1} Y_{ij,} = W_i; \qquad i = 1, \dots, (m+1) \\ \sum_{i=1}^{m+1} Y_{i,j} = w_j; \qquad j = 1, \dots, (n+1) \\ Y_{i,j} \ \in \{0, w_j\}; \ \ i = 1, \dots, (m+1) \ ; \ \ j = 1, \dots, (n+1) \end{cases}$$

Now, we change the problem to a minimization problem becomes the MKP is usually maximization. To change the BMKP can be presented it as a minimization by setting

$$K = \max_j \left\{ \frac{P_j}{w_j} \right\}$$

and subtracting the coefficients from K to obtain all the coefficients $K_{i,j}$

$$K_{m+1,j} = K; \quad K_{i,n+1} = K; \quad K_{i,j} = K - \frac{p_j}{w_j} \ ; \quad i = 1, \dots, m \ ; \quad j = 1, \dots, n$$

The formulation of MIN – MKP becomes

$$\text{MIN} - \text{MKP} \begin{cases} \min \ \mathrm{Z}_m = \sum_{i=1}^{m+1} \sum_{j=1}^{n+1} K_{i,j}\, Y_{i,j} \\ \sum_{j=1}^{n+1} Y_{i,j} = W_i; \qquad i = 1, \dots, m \\ \sum_{i=1}^{m+1} Y_{i,j} = w_j; \qquad j = 1, \dots, n \\ Y_{i,j} \in \{0, w_j\}; \quad i = 1, \dots, m\,; \ \ j = 1, \dots, n \end{cases}$$

Since we get the formulation of the $0 - 1$ multiple Knapsack Problem – Transportation we present the Adapted Transportation Algorithm for solving the problem in the following section.

5.4 Adapted Transportation Algorithm

5.4.1 Vogel Approximation Method

To find the initial solution we use the Vogel Approximation Method (VAM) described by the following steps.

1. Compute the penalty cost for each row and column; however, for each row and column, the penalty is the difference between two minimums cost.
2. Determine the largest penalty of the line to be assigned.
3. Assign the variable having the lowest unit cost.
4. Update the supply and demand.

   If there is one line remaining fill it and end

Else continue

5. Cross out the assigned column or row with zero supply or demand and return to step 1.

Remark 1. If the items are ordered such that the efficiency rate is decreasing, the variable to be assigned is the first variable of the first column of the remaining table.

Remark 2. If the items are ordered such that the efficiency rate is decreasing, the initial solution is equivalent to the greedy approximation solution (Dantzig, 1957).

5.4.2 Dual Variable and test of reduction

The dual variables u_i and v_j associated to the current solution is provided by the following system of equations

$$K_{i,j} - u_i - v_j = 0 \qquad \forall\, Y_{i,j} \in B$$

where B is the set of basic variable. Since there are $m + n$ unknown variables and $m + n - 1$ equations, by setting $u_1 = 0$ then we determine a solution of dual variables and then the reduced cost of all non-basic variables

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad ; \quad \forall\, Y_{i,j} \notin B$$

We can notice that the current solution is optimal if the reduced cost for all non-basic variables is positive i.e.

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad \geq 0.$$

Otherwise there exist at least one non-basic variable $Y_{i,j}$ such that

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \ < 0$$

In the case of the $0-1$ Multiple knapsack problem since the profit of any item j independent of the knapsack, the reduced cost of such variables verify for $i = 1, \dots, m$ and $j = 1, \dots, n$

$$Y_{i,j} = 0, \ Y_{m+1,j} > 0 \quad \text{and} \quad \widehat{K}_{i,j} = K_{i,j} - K < 0;$$

This indicates the possibility to decrease the total transportation cost. This will be tested by considering the move from $Y_{m+1,j}$ to $Y_{i,j}$; $i = 1, \dots, m$ called the j-move to $Y_{i,j}$. That move decreases the cost by

$$R_j = \widehat{K}_{i,j} \ Y_{m+1,j} \ < 0 \ ; \ i = 1, \dots, m.$$

Since the first m constraints are limited by the Knapsack capacity W_i the j-move to $Y_{i,j}$ yields the use of the surplus and some minimal necessary moves of variables

$$Y_{i,\sigma(t)} \ \text{to} \ Y_{m+1,\sigma(t)} \ ; \ t = 1, \dots, T.$$

This implies an increase of the cost equal to

$$I_j = \sum_{t=1}^{T} \widehat{K}_{m+1,\sigma(t)} \cdot X_{i,\sigma(t)} > 0; \quad i = 1, \dots, m$$

The value $R_j + I_j$ represents the change of the cost function resulting in the j-move. Therefore if $R_j + I_j \ < 0$ then the objective cost is reduced and the move retained.

Otherwise the j-move is discarded. The comparison of these successful moves progressively performed in order to retain the optimal one that is associated to the optimal solution.

In the next section, we provide illustration examples solved by using the Adapted Transportation Algorithm. One of these examples has been solved in chapter 3 by Branch and Bound Algorithm (B&B), and the second example is an illustration of the possibility to consider dependent profit unit of the item to the knapsack.

5.5 Example of illustration

5.5.1 Example 1

Consider the following multiple knapsack problem with $m = 2$ knapsacks and $n = 10$ items. The profit and weight are

$$P_j = (78, 35, 89, 36, 94, 75, 74, 79, 80, 16)$$

$$w_j = (18, 9, 23, 20, 59, 61, 70, 75, 76, 30)$$

$$W_i = (103, 156)$$

The mathematical formulation of the multiple knapsack problem is

78	35	89	36	94	75	74	79	80	16	**103**
78	35	89	36	94	75	74	79	80	16	**156**
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	

$$\begin{cases} \max\ \mathrm{Z} = \sum_{i=1}^{2} \sum_{j=1}^{10} P_j\, X_{i,j} \\ \sum_{j=1}^{10} w_j\, X_{i,j} \le W_i; \quad i = 1,2 \\ \sum_{i=1}^{2} X_{i,j} \le 1; \quad j = 1, \dots, 10 \\ X_{i,j} \in \{0,1\}; \quad i = 1,2\ ;\ j = 1, \dots, 10 \end{cases}$$

Its representation is provided by the following table

By the change of variable $Y_{i,j} = w_j X_{i,j}$ the problem becomes as following maximum transportation problem. Therefore, the following table provides the transportation problem

$\frac{39}{9}$	$\frac{35}{9}$	$\frac{89}{23}$	$\frac{9}{5}$	$\frac{94}{59}$	$\frac{75}{61}$	$\frac{37}{35}$	$\frac{79}{75}$	$\frac{20}{19}$	$\frac{8}{15}$	**103**

$\frac{39}{9}$	$\frac{35}{9}$	$\frac{89}{23}$	$\frac{9}{5}$	$\frac{94}{59}$	$\frac{75}{61}$	$\frac{37}{35}$	$\frac{79}{75}$	$\frac{20}{19}$	$\frac{8}{15}$	**156**
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	

The problem is not balanced because

$$\sum_{j=1}^{10} w_j = 441 > \sum_{i=1}^{2} W_i = 259$$

To become a balanced problem, we need to add a dummy knapsack $i = 3$ and a dummy item $n = 11$. The weight of the dummy item 11 is equal to an unknown buffer B.

$$w_{11} = B$$

The total supply dummy of knapsack 3 is equal

$$W_3 = B + \sum_{j=1}^{10} w_j - \sum_{i=1}^{2} W_i$$

$$= B + 441 - 259 = B + 182$$

Now, the problem becomes balanced

$$\sum_{i=1}^{3} W_i = 103 + 156 + (B + 441 - 259) = B + 441 = \sum_{j=1}^{11} w_j$$

The cost for the dummy knapsack and surplus items are equal zero.

The associated table is

$\frac{39}{9}$	$\frac{35}{9}$	$\frac{89}{23}$	$\frac{9}{5}$	$\frac{94}{59}$	$\frac{75}{61}$	$\frac{37}{35}$	$\frac{79}{75}$	$\frac{20}{19}$	$\frac{8}{15}$	0	**103**
$\frac{39}{9}$	$\frac{35}{9}$	$\frac{89}{23}$	$\frac{9}{5}$	$\frac{94}{59}$	$\frac{75}{61}$	$\frac{37}{35}$	$\frac{79}{75}$	$\frac{20}{19}$	$\frac{8}{15}$	0	**156**
0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{B + 182}$
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	$\boldsymbol{B}$	

By considering the highest efficiency $\frac{39}{9}$, we change the problem to a minimization problem by subtracting the coefficients from $\frac{39}{9}$. This provides after approximation the transportation problem

0	0.44	0.46	2.53	2.74	3.10	3.276	3.28	3.281	3.8	4.33	**103**
0	0.44	0.46	2.53	2.74	3.10	3.276	3.28	3.281	3.8	4.33	**156**
4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	$\boldsymbol{B + 182}$
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	$\boldsymbol{B}$	

To solve the transportation problem, we first find an initial solution by using the Vogel Approximation Method (VAM). It consists in evaluating the penalties of assigning to each row and column, a penalty which is the difference of the two lowest cost of that line. This leads to

$$p_1 = p_2 = 0.44\ ;\ p_3 = 0\ ;\ q_j = 0; j = 1, \dots, 11$$

Assignment 1. Since the largest penalties are $P_1 = P_2\ , q_j = 0,\quad j = 1, \dots, 10$ the variable to be assigned is

$$Y_{1,1} = 18 \quad \text{and} \quad S_1 = 103 - 18 = 85$$

The column 1 is crossed out from the table.

Assignment 2. Since the efficiency rate is increasing, the next variable to be assigned is

$$Y_{1,2} = 9 \quad \text{and} \quad S_2 = 85 - 9 = 76$$

The column 2 is crossed out from the table.

Assignment 3. Then the next variable to be assigned is

$$Y_{1,3} = 23 \quad \text{and} \quad S_3 = 76 - 23 = 53$$

The column 3 is crossed out from the table.

Assignment 4. Then the next variable to be assigned is

$$Y_{1,4} = 20 \quad \text{and} \quad S_4 = 53 - 20 = 33$$

The column 4 is crossed out from the table.

Since the remaining supply is lower than the weight for $Y_{1,5};\ Y_{1,6};\ Y_{1,7};\ Y_{1,8}$ and $Y_{1,9}$ then we set their coefficients equal to the largest cost

$$K_{1,5} = K_{1,6} = K_{1,7} = K_{1,8} = K_{1,9} = 4.33$$

This implies the update of their penalties to

$$q_5 = 1.59 \; ; q_6 = 1.23 \; ; q_7 = 1.054 \; ; q_8 = 1.05 \; ; q_9 = 1.049 \; ; q_{10} = 0.53 \; ; q_{11} = 0$$

Assignment 5. The largest penalty is $q_5 = 1.59$ and the variable to be assigned is

$$Y_{2,5} = 59 \quad \text{and} \quad S_5 = 156 - 59 = 97$$

The column 5 is crossed out from the table.

Assignment 6. The largest penalty is $q_6 = 1.23$ and then the variable to be assigned is

$$Y_{2,6} = 61 \qquad \text{and} \quad S_6 = 97 - 61 = 36$$

Since the remaining supply is lower than the weight for $Y_{2,7}$; $Y_{2,8}$ and $Y_{2,9}$ we set their coefficients equal to the largest cost

$$K_{2,7} = K_{2,8} = K_{2,9} = 4.33$$

The update of their penalties implies

$$q_7 = q_8 = q_9 = 0$$

Assignment 7. The largest penalty is $p_1 = p_2 = 0.53$ and then the variable to be assigned is

$$Y_{1,10} = 30 \quad \text{and} \quad S_7 = 33 - 30 = 3$$

The column 10 is crossed out from the table. Now, we fill the remaining column and row by setting

$$Y_{1,11} = 3; \quad Y_{2,11} = 36; \quad Y_{3,11} = 0; \quad Y_{3,7} = 70; \quad Y_{3,8} = 75; \quad Y_{3,9} = 76;$$

The initial feasible solution is

18	**9**	**23**	**20**						**30**	**3**
				59	**61**					**36**
						70	**75**	**76**		**0**

The total profit is

$$Z = 423 \quad \text{with} \quad B = 3 + 36 = 39$$

To find the optimal solution, we need to find the dual variables $u_i \; and \; v_j$ for all basic variables. We set $u_1 = 0$ and then we have

$$\begin{cases} u_1 + v_1 = 0 & \Rightarrow \quad v_1 = 0 \\ u_1 + v_2 = 0.44 & \Rightarrow \quad v_2 = 0.44 \\ u_1 + v_3 = 0.46 & \Rightarrow \quad v_3 = 0.46 \\ u_1 + v_4 = 2.53 & \Rightarrow \quad v_4 = 2.53 \\ u_1 + v_{10} = 3.8 & \Rightarrow \quad v_{10} = 3.8 \\ u_1 + v_{11} = 4.33 & \Rightarrow \quad v_{11} = 4.33 \end{cases}$$

Since we know $v_{11} = 4.33$; therefore, we find $u_2 = 0$. Also,

$$\begin{cases} v_5 = 2.74 \\ v_6 = 3.10 \end{cases}$$

By using $v_{11} = 4.33$, and then $u_3 = 0$. Also,

$$\begin{cases} v_7 = 4.33 \\ v_8 = 4.33 \\ v_9 = 4.33 \end{cases}$$

After that, we need to find the cost of all non – basic variables

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad ; \quad \forall\, Y_{i,j} \notin B$$

The reduced costs for the non – basic variables in the first row are

$$\widehat{K}_{1,5} = K_{1,5} - u_1 - v_5 \quad \Rightarrow \quad 2.74 - 0 - 2.27 = 0.47$$

$$\widehat{K}_{1,6} = K_{1,6} - u_1 - v_6 \quad \Rightarrow \quad 3.10 - 0 - 3.10 = 0$$

$$\widehat{K}_{1,7} = K_{1,7} - u_1 - v_7 \Rightarrow 3.276 - 0 - 4.33 = -1.054$$

$$\widehat{K}_{1,8} = K_{1,8} - u_1 - v_8 \quad \Rightarrow 3.28 - 0 - 4.33 = -1.05$$

$$\widehat{K}_{1,9} = K_{1,9} - u_1 - v_9 \quad \Rightarrow 3.281 - 0 - 4.33 = -1.049$$

For the second row

$$\widehat{K}_{2,1} = K_{2,1} - u_2 - v_1 \quad \Rightarrow \quad 0 - 0 - 0 = 0$$

$$\widehat{K}_{2,2} = K_{2,2} - u_2 - v_2 \quad \Rightarrow \quad 0.44 - 0 - 0.44 = 0$$

$$\widehat{K}_{2,3} = K_{2,3} - u_2 - v_3 \quad \Rightarrow \quad 0.46 - 0 - 0.46 = 0$$

$$\widehat{K}_{2,4} = K_{2,4} - u_2 - v_4 \quad \Rightarrow 2.53 - 0 - 2.53 = 0$$

$$\widehat{K}_{2,7} = K_{2,7} - u_2 - v_7 \quad \Rightarrow 3.276 - 0 - 4.33 = -1.054$$

$$\widehat{K}_{2,8} = K_{2,8} - u_2 - v_8 \quad \Rightarrow 3.28 - 0 - 4.33 = -1.05$$

$$\widehat{K}_{2,9} = K_{2,9} - u_2 - v_9 \quad \Rightarrow 3.281 - 0 - 4.33 = -1.049$$

$$\widehat{K}_{2,10} = K_{2,10} - u_2 - v_{10} \quad \Rightarrow 3.8 - 0 - 3.8 = 0$$

The reduced cost for the third row

$$\widehat{K}_{3,1} = K_{3,1} - u_3 - v_1 \quad \Rightarrow \quad 4.33 - 0 - 0 = 4.33$$

$$\widehat{K}_{3,2} = K_{3,2} - u_3 - v_2 \quad \Rightarrow 4.33 - 0 - 0.44 = 3.89$$

$$\widehat{K}_{3,3} = K_{3,3} - u_3 - v_3 \quad \Rightarrow \quad 4.33 - 0 - 0.46 = 3.87$$

$$\widehat{K}_{3,4} = K_{3,4} - u_3 - v_4 \quad \Rightarrow \quad 4.33 - 0 - 2.53 = 1.8$$

$$\widehat{K}_{3,5} = K_{3,5} - u_3 - v_5 \quad \Rightarrow 4.33 - 0 - 2.74 = 1.59$$

$$\widehat{K}_{36} = K_{3,6} - u_3 - v_6 \quad \Rightarrow 4.33 - 0 - 3.10 = 1.23$$

$$\widehat{K}_{3,10} = K_{3,10} - u_3 - v_{10} \quad \Rightarrow \quad 4.33 - 0 - 3.8 = 0.53$$

The current solution with reduced cost is presented below

18	**9**	**23**	**20**	0.47	0	−1.054	−1.05	−1.049	**30**	**3**
0	0	0	0	**59**	**61**	−1.054	−1.05	−1.049	0	**36**
4.33	3.89	3.87	1.8	1.59	1.23	**70**	**75**	**76**	0.53	**0**

Since the least value is $\widehat{K}_{2,7} = -1.054$; therefore, it indicates a possibility of reducing the cost. The variable $Y_{2,7}$ is reducing the basic. This implies the moving of $Y_{3,7}$ to $Y_{2,7}$. This implies the reduction cost

$$R_7 = 70 \times (-1.054) = -73.78$$

It also implies the move of the weight $w_6 = 61$ to $Y_{3,6}$. This implies an increasing cost

$$I_7 = 61 \times (1.23) = 75.03$$

It also implies the possible move of $w_6 = 61$ to $Y_{1,6}$. This implies an increasing cost

$$I_7 = 9 \times (3.89) + 30 \times (0.53) = 50.91$$

By adding $R_7 + I_7$ " the move of $w_6 = 61$ to $Y_{3,6}$" the total becomes $1.25 > 0$; therefore, this moving from $Y_{3,6}$ to $Y_{2,6}$ is not acceptable to reduce the current cost z. By adding $R_7 + I_7$ " the move of $w_6 = 61$ to $Y_{1,6}$" the total becomes $-22.87 < 0$ then the move is maintained.

Next, the least value is $\widehat{K}_{1,7} = -1.054$; therefore, it indicates a possibility of reducing the cost. The variable $Y_{1,7}$ is reducing the basic. This implies the moving of $Y_{3,7}$ to $Y_{1,7}$

This implies the reduction cost

$$R_7 = 70 \times (-1.054) = -73.78$$

It also implies the moving of $Y_{1,4}$ and $Y_{1,10}$, to respectively $X_{3,4}$and $X_{3,10}$. This implies an increasing cost

$$I_7 = 20 \times (1.8) + 30 \times (0.53) = 51.9$$

By adding them the total becomes $-21.88 < 0$ then the move of $Y_{1,7}$ is maintained.

The third least value is $\widehat{K}_{2,8} = -1.05$; therefore, it indicates a possibility of reducing the cost. The variable $X_{2,8}$ is reducing the basic. This implies the moving of $Y_{3,8}$to $Y_{2,8}$. This implies the reduction cost

$$R_8 = 75 \times (-1.05) = -78.75$$

It also implies the move of the weight $w_6 = 61$ to $Y_{3,6}$. This implies an increasing cost

$$I_8 = 61 \times (1.23) = 75.03$$

By adding them the result is decreasing $-3.72 < 0$ then the move is maintained.

It also implies the possible move of $w_6 = 61$ to $Y_{1,6}$. This implies an increasing cost

$$I_8 = 9 \times (3.89) + 30 \times (0.53) = 50.91$$

By adding them the result is decreasing $-27.84 < 0$ then the move is maintained

Next, the least value is $\widehat{K}_{1,8} = -1.05$; therefore, it shows a possibility of reducing the cost.

The variable $Y_{1,8}$ is reducing the basic. This implies the moving of $Y_{1,4}$ and $Y_{1,10}$ to respectively $Y_{3,4}$ and $Y_{3,10}$. This implies the reduction cost $R_8 = -78.75$.

It also implies an increasing cost

$$I_8 = 20 \times (1.8) + 30 \times (0.53) = 51.9$$

By adding them, it is given a decreasing cost $-26.85 < 0$. That means the moving from $Y_{3,8}$ to $Y_{1,8}$ is maintained.

Since we still have negative values, so the least value is $\widehat{K}_{2,9} = -1.049$; therefore, there is a possibility of reducing the cost. The variable $Y_{2,9}$ is reducing the basic. This implies the moving of $Y_{3,9}$ to $X_{2,9}$. This implies the reduction cost

$$R_9 = 76 \times (-1.049) = -79.496$$

It also implies the move of the weight $w_6 = 61$ to $Y_{3,6}$. This implies an increasing cost

$$I_9 = 61 \times (1.23) = 75.03$$

By adding them, the result becomes decreasing $-3.924 < 0$ then the move is maintained.

It also implies the move of the weight $w_6 = 61$ to $Y_{1,6}$. This implies an increasing cost

$$I_9 = 20 \times (1.8) + 30 \times (0.53) = 51.9$$

By adding them, the result becomes decreasing $-28.814 < 0$ then the move is maintained.

The last least value is $\widehat{K}_{1,9} = -1.049$; therefore, it indicates a possibility of reducing the cost. The variable $Y_{1,9}$ is reducing the basic. This implies the moving of $Y_{1,4}$ and $Y_{1,10}$ to respectively $Y_{3,4}$ and $Y_{3,10}$. This implies the reduction cost $R_9 = -79.496$.

It also implies an increasing cost

$$I_9 = 20 \times (1.8) + 30 \times (0.53) = 51.9$$

By adding them the result is increasing $-27.824 < 0$ then the move is maintained.

Since we have some moves are maintained we choose the best move which is given more reduced cost. Therefore, the best moving is form Y_{39} to Y_{29} because the reducing cost is equal $-28.814 < 0$.

The current solution becomes

18	+	**23**	0	0	**61**	+	+	0	+	**1**
0	+	0	**20**	**59**	0	+	+	**76**	+	**1**
4.33	**9**	3.87	1.8	1.59	1.23	**70**	**75**	1.049	**30**	**37**

The total of maximum profit is equal $z = 452$ with $B = 39$

5.5.2 Example 2

Consider the following Multiple Knapsack Problem having

$m = 2$ knapsack and $n = 10$ items. The unit profit and weight are

$$P_{i,j} = \begin{pmatrix} 78, 35, 89, 36, 94, 75, 74, 79, 80, 16 \\ 74, 32, 92, 34, 92, 78, 72, 80, 78, 20 \end{pmatrix}$$

$$w_j = (18, 09, 23, 20, 59, 61, 70, 75, 76, 30)$$

$$W_1 = 103, W_2 = 156$$

The multiple Knapsack Problem formulation is

$$\begin{cases} \max Z = \sum_{i=1}^{2}\sum_{j=1}^{10} P_{i,j}X_{i,j} \\ \sum_{j=1}^{10} w_j X_{i,j} \le W_i\,; \;\; i = 1,2 \\ \sum_{i=1}^{2} X_{i,j} = 1\,; \;\; j = 1,\dots,10 \\ X_{i,j} \in \{0,1\}; \;\; i = 1,2 \;\; ; j = 1,\dots,10 \end{cases}$$

The matrix costs are provided by the following table

78	35	89	32	94	75	74	79	80	16	**103**
74	32	92	34	92	78	72	80	78	20	**156**
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	

By the change of variable $Y_{i,j} = w_j\, X_{i,j}$ the problem

The transportation problem is provided by the following table

$\frac{39}{9}$	$\frac{35}{9}$	$\frac{89}{23}$	$\frac{8}{5}$	$\frac{94}{59}$	$\frac{75}{61}$	$\frac{37}{35}$	$\frac{79}{75}$	$\frac{20}{19}$	$\frac{8}{15}$	**103**
$\frac{2}{9}$	$\frac{32}{9}$	$\frac{92}{23}$	$\frac{17}{10}$	$\frac{92}{59}$	$\frac{78}{61}$	$\frac{36}{35}$	$\frac{16}{15}$	$\frac{39}{38}$	$\frac{2}{3}$	**156**
18	9	23	20	59	61	70	75	76	30	

The problem is not balanced because

$$\sum_{j=1}^{10} w_j = 441 > \sum_{i=1}^{2} W_i = 259$$

Since the problem is not balanced we need to add a dummy knapsack $i = 3$ and a dummy item $j = 11$.

The weight of the dummy item 11 is equal to an unknown buffer B.

$$w_{11} = B$$

The total supply of surplus knapsack 3 is equal

$$W_{(m+1)} = B + \sum_{j=1}^{n} w_j - \sum_{i=1}^{m} W_i$$

$$= B + 441 - 259 = B + 182$$

Now, the problem becomes balanced because

$$\sum_{i=1}^{3} W_i = 103 + 156 + (B + 441 - 259) = B + 441 = \sum_{j=1}^{11} w_j$$

The cost for the dummy knapsack and dummy item are equal zero.

The associated table is

78	35	89	32	94	75	74	79	80	16	0	**103**
74	32	92	34	92	78	72	80	78	20	0	**156**
0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{B + 182}$
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	$\boldsymbol{B}$	

By considering the highest efficiency $\frac{39}{9}$, we can change the problem a minimization by subtracting the coefficients from $\frac{39}{9}$. This provides often approximation the transportation problem

0	0.44	0.46	2.73	2.74	3.10	3.276	3.28	3.281	3.8	4.33	**103**
0.22	0.77	0.33	2.63	2.77	3.055	3.305	3.27	3.307	3.67	4.33	**156**
4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	4.33	$\boldsymbol{182 + B}$
18	**9**	**23**	**20**	**59**	**61**	**70**	**75**	**76**	**30**	**B**	

To solve the transportation problem, we first find an initial solution by using the Vogel Approximation Method (VAM). It consists in evaluating the penalties of assigning to each row and column, a penalty which is the difference of the two lowest cost of that line. This leads to

$$p_1 = 0.44\ ;\ p_2 = 0.11\ ;\ p_3 = 0\ ;\ q_1 = 0.22\ ;\ q_2 = 0.33\ ;\ q_3 = 0.13\ ;\ q_4 = 0.1\ ;$$

$q_5 = 0.03\ ;\ q_6 = 0.045\ ;\ q_7 = 0.029\ ;\ q_8 = 0.01\ ;\ q_9 = 0.026\ ;\ q_{10} = 0.13$ and $q_{11} = 0$

Assignment 1. Since the largest penalty is $P_1 = 0.44\ , q_j = 0,\ j = 1, \dots, 10$, the variable to be assigned is

$$Y_{1,1} = 18 \quad \text{and} \quad S_1 = 103 - 18 = 85$$

The column 1 is crossed out from the table.

Assignment 2. Since the largest penalty is $P_2 = 0.44, q_j = 0,\ j = 1, \dots, 10$ the variable to be assigned is

$$Y_{2,3} = 23 \quad \text{and} \quad S_2 = 156 - 23 = 133$$

The column 3 is crossed out from the table.

Assignment 3. Since the largest penalty is $P_2 = 2.29,\ q_j = 0,\ j = 1, \dots, 10$; therefore, the variable to be assigned is

$$Y_{1,2} = 9 \quad \text{and} \quad S_3 = 85 - 9 = 76$$

The column 2 is crossed out from the table.

Assignment 4. Since the largest penalties are $p_2 = 0.14,\ \ j = 1, \dots, 10$ the variable to be assigned is

$$X_{2,4} = 20 \quad \text{and} \quad S_4 = 133 - 20 = 113$$

The column 4 is crossed out from the table.

Assignment 5. The largest penalty is $p_1 = 0.36$ and the variable to be assigned is

$$Y_{1,5} = 59 \quad \text{and} \quad S_5 = 76 - 59 = 17$$

The column 5 is crossed out from the table.

Since the remaining supply is lower than the weight for $Y_{1,6};\ Y_{1,7};\ Y_{1,8}\ ;\ Y_{1,9}$ and $Y_{1,10}$

then we set their coefficients equal to the largest cost

$$K_{1,6} = 4.33; \quad K_{1,7} = 4.33; \quad K_{1,8} = 4.33\ ; \quad K_{1,9} = 4.33; \quad K_{1,10} = 4.33$$

This implies the update of their penalties to

$$q_6 = 1.275; \quad q_7 = 1.025; \quad q_8 = 1.06; \quad q_9 = 1.023; \quad q_{10} = 0.66; \text{ and } q_{11} = 0$$

Assignment 6. The largest penalty is $q_6 = 1.275$ and then the variable to be assigned is

$$Y_{2,6} = 61 \qquad \text{and} \quad S_6 = 113 - 61 = 52$$

The column 6 is crossed out from the table.

Since the remaining supply is lower than the weight for $Y_{2,7};\ Y_{2,8}$, and $Y_{2,9}$ then we set their coefficients equal to the largest cost

$$K_{2,7} = 4.33; \quad K_{2,8} = 4.33; \quad K_{2,9} = 4.33$$

The update of their penalties implies

$$q_7 = q_8 = q_9 = 0$$

Assignment 7. The largest penalty is $P_2 = 0.66$ and then the variable to be assigned is

$$Y_{2,10} = 30 \qquad w_{10} = 0 \quad \text{and} \quad S_7 = 52 - 30 = 22$$

The column 10 is crossed out from the table. Now, we fill the remaining column and row by setting

$$Y_{1,11} = 17\,;\;\; Y_{2,11} = 22;\quad Y_{3,11} = 0;\;\; Y_{3,7} = 70;\quad Y_{3,8} = 75;\quad Y_{3,9} = 76$$

The table of initial solution is

18	**9**			**59**						**17**
		23	**20**		**61**				**30**	**22**
						70	**75**	**76**		**0**

The total profit is

$$Z = 431 \quad \text{with} \qquad B = 17 + 22 = 39$$

To find the optimal solution, we need to find the dual variables $u_i \; and \; v_j$ for all basic variables.

We set $u_1 = 0$ and then we have

$$\begin{cases} u_1 + v_1 = 0 & \Rightarrow \quad v_1 = 0 \\ u_1 + v_2 = 0.44 & \Rightarrow \; v_2 = 0.44 \\ u_1 + v_5 = 2.74 & \Rightarrow \quad v_5 = 2.74 \\ u_1 + v_{11} = 4.33 & \Rightarrow \quad v_{11} = 4.33 \end{cases}$$

Since we know $v_{11} = 4.33$; therefore, we find $u_2 = 0$. Also,

$$\begin{cases} v_3 = 0.33 \\ v_4 = 2.63 \\ v_6 = 3.055 \\ v_{10} = 3.67 \end{cases}$$

By using $v_{11} = 4.33$ and then $u_3 = 0$. Also,

$$\begin{cases} v_7 = 4.33 \\ v_8 = 4.33 \\ v_9 = 4.33 \end{cases}$$

After that, we need to find the reduced cost of all non – basic variables

$$\widehat{K}_{i,j} = K_{i,j} - u_i - v_j \quad ; \quad \forall Y_{i,j} \notin B$$

The reduced costs for the non – basic variables in the first row are

$$\widehat{K}_{1,3} = K_{1,3} - u_1 - v_3 \quad \Rightarrow \quad 0.46 - 0 - 0.33 = 0.13$$

$$\widehat{K}_{1,4} = K_{1,4} - u_1 - v_4 \quad \Rightarrow \quad 2.73 - 0 - 2.63 = 0.1$$

$$\widehat{K}_{1,6} = K_{1,6} - u_1 - v_6 \Rightarrow 3.10 - 0 - 3.055 = 0.045$$

$$\widehat{K}_{1,7} = K_{1,7} - u_1 - v_7 \quad \Rightarrow 3.276 - 0 - 4.33 = -1.054$$

$$\widehat{K}_{1,8} = K_{1,8} - u_1 - v_8 \quad \Rightarrow 3.28 - 0 - 4.33 = -1.05$$

$$\widehat{K}_{1,9} = K_{1,9} - u_1 - v_9 \quad \Rightarrow 3.281 - 0 - 4.33 = -1.049$$

$$\widehat{K}_{1,10} = K_{1,10} - u_1 - v_{10} \quad \Rightarrow 3.8 - 0 - 3.67 = 0.13$$

For the second row

$$\widehat{K}_{2,1} = K_{2,1} - u_2 - v_1 \quad \Rightarrow \quad 0.22 - 0 - 0 = 0.22$$

$$\widehat{K}_{2,2} = K_{2,2} - u_2 - v_2 \quad \Rightarrow \quad 0.77 - 0 - 0.44 = 0.33$$

$$\widehat{K}_{2,5} = K_{2,5} - u_2 - v_5 \quad \Rightarrow \quad 2.77 - 0 - 2.47 = 0.03$$

$$\widehat{K}_{2,7} = K_{2,7} - u_2 - v_7 \quad \Rightarrow 3.305 - 0 - 4.33 = -1.025$$

$$\widehat{K}_{2,8} = K_{2,8} - u_2 - v_8 \quad \Rightarrow \ 3.27 - 0 - 4.33 = -1.06$$

$$\widehat{K}_{2,9} = K_{2,9} - u_2 - v_9 \quad \Rightarrow \ 3.307 - 0 - 4.33 = -1.023$$

The reduced cost for the third row

$$\widehat{K}_{3,1} = K_{3,1} - u_3 - v_1 \quad \Rightarrow \quad 4.33 - 0 - 0 = 4.33$$

$$\widehat{K}_{3,2} = K_{3,2} - u_3 - v_2 \quad \Rightarrow \quad 4.33 - 0 - 0.44 = 3.89$$

$$\widehat{K}_{3,3} = K_{3,3} - u_3 - v_3 \quad \Rightarrow \quad 4.33 - 0 - 0.33 = 4$$

$$\widehat{K}_{3,4} = K_{3,4} - u_3 - v_4 \quad \Rightarrow \quad 4.33 - 0 - 2.63 = 1.7$$

$$\widehat{K}_{3,5} = K_{3,5} - u_3 - v_5 \quad \Rightarrow \quad 4.33 - 0 - 2.74 = 1.59$$

$$\widehat{K}_{36} = K_{3,6} - u_3 - v_6 \quad \Rightarrow \ 4.33 - 0 - 3.055 = 1.275$$

$$\widehat{K}_{3,10} = K_{3,10} - u_3 - v_{10} \quad \Rightarrow \quad 4.33 - 0 - 3.67 = 0.66$$

The reduced cost for non-basic variables with current solution are represented by the

following table:

18	**9**	0.13	0.1	**59**	0.045	−1.054	−1.05	−1.049	0.13	**17**
0.22	0.33	**23**	**20**	0.03	**61**	−1.025	−1.06	−1.023	**30**	**22**
4.33	3.89	4	1.7	1.59	1.275	**70**	**75**	**76**	0.66	**0**

Since the most negative value is $\widehat{K}_{2,8} = -1.06$, then the move of $Y_{3,8} = 75$ to $Y_{2,8}$ will reduce the objective by

$$R_8 = 75 \times (-1.06) = -79.5$$

This will force at least the move of the weight $w_6 = 61$ to $Y_{3,6}$

$$I_8 = 61 \times (1.275) = 77.775$$

By adding them, the total becomes $-1.725 < 0$ then the move is maintained. It also implies the possible move of the weight $w_6 = 61$ to $Y_{1,6}$ and that will force at least the move of $Y_{1,2} = 9$ to $Y_{3,2}$ and $Y_{2,10} = 30$ to $Y_{3,10}$. This implies an increasing

$$I_8 = 9 \times (3.89) + 30 \times (0.66) = 54.8$$

Since adding R_8 and I_8 the total is $-24.69 < 0$, then the move of $Y_{3,8} = 75$ is maintained.

Next the least negative value is $\widehat{K}_{1,7} = -1.054$, and then the move of $Y_{3,7} = 70$ to $Y_{1,7}$ will reduce the objective by

$$R_7 = 70 \times (-1.054) = -73.78$$

This will force at least the move of $Y_{2,4} = 20$ to $Y_{3,4}$ and $Y_{2,10} = 30$ to $Y_{3,11}$

$$I_7 = 20 \times (1.7) + 30 \times (0.66) = 53.8$$

By adding R_7 and I_7 the total becomes $-19.98 < 0$, then the move of $Y_{3,7} = 70$ is maintained.

Since the next least reduced cost is $\widehat{K}_{1,8} = -1.05$, then the move of $Y_{3,8} = 75$ to $Y_{1,8}$ will reduce the objective by

$$R_8 = 75 \times (-1.05) = -78.75$$

This will force at least the move of $Y_{2,4} = 20$ to $Y_{3,4}$ and $Y_{2,10} = 30$ to $Y_{3,11}$

$$I_8 = 20 \times (1.7) + 30 \times (0.66) = 53.8$$

Since adding R_8 and I_8 the total is $-24.95 < 0$, then the move of $Y_{3,8} = 75$ is maintained.

Since the next least reduced cost is $\widehat{K}_{1,9} = -1.049$, then the move of $Y_{3,9} = 76$ to $Y_{1,9}$ will reduce the objective by

$$R_9 = 76 \times (-1.049) = -79.724$$

This will force at least the move of $Y_{2,4} = 20$ to $Y_{3,4}$ and $Y_{2,10} = 30$ to $Y_{3,11}$

$$I_9 = 20 \times (1.7) + 30 \times (0.66) = 53.8$$

Since adding R_9 and I_9 equal $-25.924 < 0$, then the move of $Y_{3,9} = 76$ is maintained.

Since the most negative value is $\widehat{K}_{2,7} = -1.025$, then the move of $Y_{3,7} = 70$ to $Y_{2,7}$ which reduce the objective by

$$R_7 = 70 \times (-1.025) = -71.75$$

This will force at least the move of the weight $w_6 = 61$ to $Y_{3,6}$

$$I_7 = 61 \times (1.275) = 77.775$$

By adding them, the total becomes $6.025 > 0$ then the move is not improving and is discarded.

It also implies the possible move of the weight $w_6 = 61$ to $Y_{1,6}$ and that will force at least the move of $Y_{1,2} = 9$ to $Y_{3,2}$ and $Y_{2,10} = 30$ to $Y_{3,10}$. This implies an increasing

$$I_7 = 9 \times (3.89) + 30 \times (0.66) = 54.8$$

Since adding R_7 and I_7 the total is $-17.95 < 0$, then the move of $Y_{3,7} = 70$ is maintained.

Since the last least negative value is $\widehat{K}_{2,9} = -1.023$, then the move of $Y_{3,9} = 76$ to $Y_{2,9}$ which reduce the objective by

$$R_9 = 76 \times (-1.023) = -77.748$$

This will force at least the move of the weight $w_6 = 61$ to $Y_{3,6}$

$$I_9 = 61 \times (1.275) = 77.775$$

By adding them, the total becomes $0.027 > 0$ then the move is not improving and is discarded.

It also implies the possible move of the weight $w_6 = 61$ to $Y_{1,6}$ and that will force at least the move of $Y_{1,2} = 9$ to $Y_{3,2}$ and $Y_{2,10} = 30$ to $Y_{3,10}$. This implies an increasing

$$I_9 = 9 \times (3.89) + 30 \times (0.66) = 54.8$$

Since adding R_9 and I_9 the total is $-22.948 < 0$, then the move of $Y_{3,9} = 76$ is maintained.

Since we have some moves are maintained we choose the best move which is given more reduced cost. Therefore, the best moving is form $Y_{3,9} = 76$ to $Y_{1,9}$ because the reducing cost is equal $-25.924 < 0$.

The current solution becomes

18	**9**	0.13	+	+	0.045	+	+	**76**	+	**0**
0.22	0.33	**23**	+	**59**	**61**	+	+	+	+	**13**
4.33	3.89	4	**20**	+	1.275	**70**	**75**	+	**30**	**26**

The maximum profit is equal $Z = 446$ with $B = 39$.

5.6 Conclusion

In this chapter, we have presented the Multiple Knapsack Problem (MKP) as a Linear Transportation Problem and provide a new approach for solving $0 - 1$ Multiple

Knapsack Problem (MKP). To find an initial solution Vogel's Approximation Method (VAM) is used, and then an Adapted Transportation Algorithm (ATA) is applied to find an optimal solution. We show that by providing two different examples: the first one, the profit unit depends on the knapsack and second example, the profit unit depends on the item to the knapsack. In addition, the approach can be extended to some variants of the Multiple Knapsack Problem such as the multiple Subset-sum problem.

Chapter 6

6 Multiple Assignment Problem

6.1 Introduction

The Multiple Assignment Problem (MAP) is defined to assign multiple tasks to a group of agents so that all the capacity constraints should be satisfied, and the total cost is minimized [67]. A task can be assigned to more than one agent, and one agent can have more than one task assigned to him. Therefore, the MAP is a generalized of the Assignment problem (AP). The MAP is a group-matching problem while the Assignment Problem is a one to one matching problem.

6.2 Mathematical Formulation of the MAP

The Multiple Assignment Problem (MAP) consists in assigning among n tasks to m agents in order to minimize a resulting total cost. To formulate the Multiple Assignment Problem, we consider the following parameters.

$P_{i,j}$: Cost of assigning task j to agent i.

W_i: Capacity of task to be assigned to agent i.

V_j: Capacity of agent to be assigned to task j.

The decision variables are

$$X_{i,j} = \begin{cases} 1 & \text{if task } i \text{ is assigned to } j \\ 0 & \text{else} \end{cases}$$

This implies the following formulation of the Multiple Assignment Problem (MAP)

$$MAP \begin{cases} \min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} X_{i,j} \\ \sum_{j=1}^{n} X_{i,j} \leq W_i ; \quad i = 1, \dots, m \\ \sum_{i=1}^{m} X_{i,j} \leq V_j ; \quad j = 1, \dots, n \\ X_{i,j} = \{0, 1\} ; \quad i = 1, \dots, m; \quad j = 1, \dots, n \end{cases}$$

The objective function represents the minimization of the total cost of assigning tasks to agents. First set of constraints represents the sum of assigning task j to agent should not exceed the capacity W_i. For the third equation, it says the sum of assigning agent i to task should not exceed the capacity V_j.

The MAP was introduced by Walkup and MacLaren in 1964 on a more general aspect [67] where the unit cost $P_{i,j}$ were reduced for a qualification matrix. Since it does not seem to attract more research. In addition, they provided three particular cases of the Multiple Assignment Problem.

First, when $m = n$, each task is assigned to only one agent, and each agent performs only one task then the Multiple Assignment Problem (MAP) becomes an Assignment Problem (AP).

$$AP \begin{cases} \min Z = \sum_{i=1}^{n}\sum_{j=1}^{n} P_{i,j}\, X_{i,j} \\ \sum_{i=1}^{n} X_{i,j} = 1; \quad j = 1, \dots, n \\ \sum_{j=1}^{n} X_{i,j} = 1\,; \quad i = 1, \dots, n \\ X_{i,j} = \{0, 1\}; \quad i, j = 1, \dots, n \end{cases}$$

Second case, when each agent performs only one task, the problem becomes as

$$\begin{cases} \min Z = \sum_{i=1}^{m}\sum_{j=1}^{n} P_{i,j}\, X_{i,j} \\ \sum_{i=1}^{m} X_{i,j} \le V_j\,; \;\; j = 1, \dots, n \\ \sum_{j=1}^{n} X_{i,j} = 1\,; \quad i = 1, \dots, m \\ X_{i,j} = \{0, 1\}; \quad i = 1, \dots, m; \;\; j = 1, \dots, n \end{cases}$$

Walkup and Maclaren [67] provided an example for this situation. The problem is to find a best assignment of agent to prioritized tasks where task i has a quota of agent. Finally, when each task is assigned to only one agent, the problem becomes as

$$\begin{cases} \min Z = \sum_{i=1}^{m}\sum_{j=1}^{n} P_{i,j}\, X_{i,j} \\ \sum_{i=1}^{m} X_{i,j} = 1\,; \quad j = 1, \dots, n \\ \sum_{j=1}^{n} X_{i,j} \le W_i\,; \quad i = 1, \dots, m \\ X_{i,j} = \{0, 1\}\,; \quad i = 1, \dots, m; \;\; j = 1, \dots, n \end{cases}$$

Walkup and Maclaren [66] presented an example which is called a target – assignment problem and introduced by Manne [39]. They showed that the target – assignment problem is a special case of the Multiple Assignment problem (MAP). Moreover, the last two cases are considered as special cases of the Generalized Assignment Problem with $w_{i,j} = 1$. They are related to the Group Role Assignment Problem (GRAP) introduced with more details in the next section.

6.3 Group Role Assignment Problem

It is defined to select among n roles to m agents in order to maximize a resulting total profit. The unit profit is represented by $P_{i,j}$. The role can be assigned to more than one agent and the agent can receive only one role, the formulation of (GRAP) is

$$GRAP \begin{cases} \max Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} X_{i,j} \\ \sum_{i=1}^{m} X_{i,j} \leq V_j; \;\; j = 1, \dots, n \\ \sum_{j=1}^{n} X_{i,j} = 1; \quad i = 1, \dots, m \\ X_{i,j} = \{0, 1\}; \quad i = 1, \dots, m; \;\; j = 1, \dots, n \end{cases}$$

One difference is on the objective function indeed the Group Role Assignment Problem is usually formulated as a maximization problem which can be early transformed as minimization. So defined, the GRAP is a Multiple Knapsack Problem where the weight $w_i = 1$ and $P_{i,j}$ dependant of i and j. For this reason, the GRAP is more indicated to the GAP problem where the weights

$$w_{i,j} = 1 \; ; i = 1, \dots, m \; ; j = 1, \dots, n$$

The weights are uniform for all the agents. In that meaning it is a Uniform Generalized Assignment Problem. Since the group role assignment problem is known as a hard problem because it needs advanced methodologies for example information classification, data mining, pattern search and matching [76]

This problem has been proposed by Zhu, Zhou and Alkins (2012) and introduced an efficient algorithm based on the Kuhn – Munkers (KM) Algorithm [76]. They worked to show the Group Role Assignment Problem can be converted to the Generalized Assignment Problem (GAP). In deed they prove that by providing a numerical example and analyse the solution' performances [76].

Role assignment (RA) is defined to be a scientific task in role-based collaboration [76]. Since it consists of three categories: agent evaluation, group role assignment and role transfer, where the group role assignment is a time – consuming process [76].

Role Based Collaboration (RBC) is known as an emerging methodology to facilitate an organization structure, to provide orderly system behaviour, and to consolidate system security for both human and nonhuman entities that collaborate and coordinate their activities with or within system [76] [75]. It consists to three main tasks: role negotiation, assignment, and execution [75].

For that, the link between Role Assignment (RA) and Role – Based Collaboration (RBC) is a Role Assignment critical aspect of Role – Based Collaboration since (RA) mostly

affects the efficiency of collaboration and the degree of happiness among members involved in the collaboration [76].

As we said before role assignment has three steps: agent evaluation, group role assignment, and role transfer. In addition, qualifications are the basic requirement for role – related activities [6]. Agent evaluation relies on three indexes: capacities, experiences and credits of agents based on role specifications [76].

It starts a group by assigning roles to agents to accomplish the highest achievement. Role transfer is also can name dynamic role assignment and defines to transport agent role to meet the demand of the system changes [76].

6.4 Real world application of the GRAP

The group role assignment has been applied to a real world problem. Since Zhu, Zhou and Alkins [76] implement the Group Role Assignment Problem (GRAP) by a soccer team. A soccer team consists of 20 players and are represented by ($a_0 - a_{19}$) in total. Moreover, they partition the team into four parts and the team consists of 11 players. The team in the field plays as $4 - 3 - 3$ and is represented by the roles r_0, r_1, r_2 and r_3 where one goalkeeper (r_0), four backs (r_1), three midfields, and three forwards (r_3). Figure 1 represents the 20 players and the 4 roles.

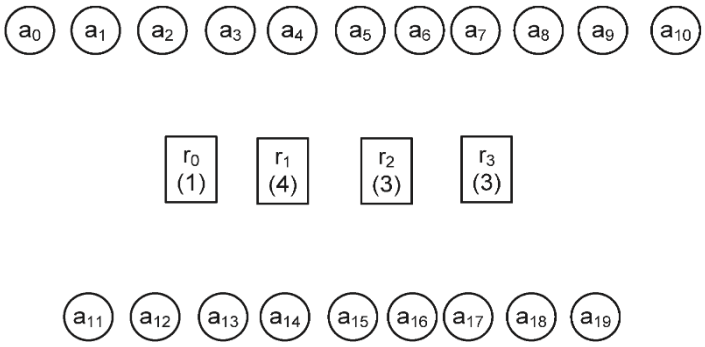

Figure 3: Soccer Team [76]

The first group is $j = 0$ with $V_0 = 1$ only one goalkeeper is necessary in a team.

The second group is $j = 1$ with $V_1 = 4$. The system of the team needs four defenders to be selected. The third group is $j = 2$ with $V_2 = 3$. The system of the team needs three middle players to be selected. The fourth group is $j = 3$ with $V_3 = 3$. The system of the team needs three offensive players to be selected. The formulation of the group Role Assignment problem becomes

$$GRAP \begin{cases} \text{Max}\, Z = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j}\, X_{i,j} \\ \sum_{i=1}^{m} X_{i,0} = 1 \;\; ; \;\; (j = 0) \\ \sum_{i=1}^{m} X_{i,1} = 4 \;\; ; \;\; (j = 1) \\ \sum_{i=1}^{m} X_{i,2} = 3 \;\; ; \;\; (j = 2) \\ \sum_{i=1}^{m} X_{i,3} = 3 \;\; ; \;\; (j = 3) \\ \sum_{j=1}^{n} X_{i,j} \leq 1 \;\; ; \;\; i = 0, \dots, 19 \\ X_{i,j} = \{0, 1\}; \;\; i = 0, \dots, 19; \;\; j = 0, \dots, 3 \end{cases}$$

The most critical task of the coach is to determine 11 players to be on the field.

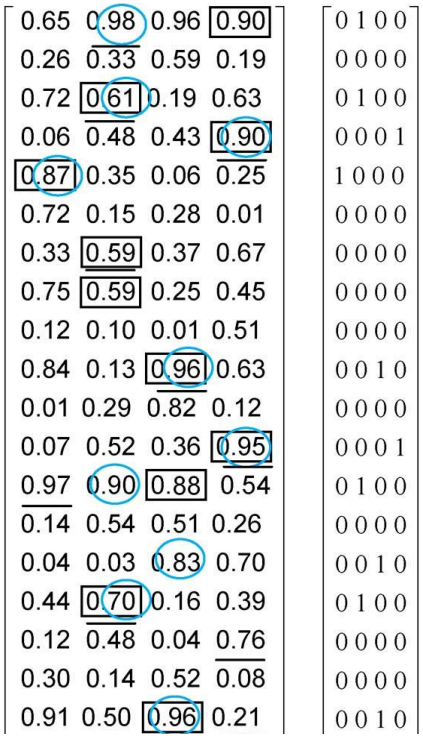

Figure 4: Evaluation values of agents and roles and the assignment matrix [76]

The data in figure 4 represents the evaluation values of players with respect to each role. Since rows refer to the players and columns indicate to roles and these values reverse the individual performance of each player related to a specific location. The aim of this problem is to improve the whole team's performance via preparing role assignment. Therefore, to make this problem simple, they suppose the team performance as a simple sum of the selected player's performance on their roles. For that, the coach has used several strategies to find an exact solution. For instance of the strategy is from $r_0\ to\ r_3$ to

exclusive the best players where they have not chosen yet, i.e., he selects the underlined digits. The total sum is 9.23 showing in the first row of the table below.

Figure 3 shows an Implementation all the strategies; they obtain all the other 22 permutations.

Strategy	Assignment for {r₀}{r₁}{r₂}{r₃} Note: In the curly braces are the numbers of players.	Group Performance
(r_0, r_1, r_2, r_3)	{12}{0, 2, 6, 15}{9, 18, 19}{3, 11, 16}	9.23
(r_0, r_1, r_3, r_2)	{12}{0, 2, 6, 15}{9, 14, 18}{3, 11, 19}	9.30
(r_0, r_2, r_1, r_3)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_0, r_2, r_3, r_1)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_0, r_3, r_1, r_2)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_0, r_3, r_2, r_1)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_1, r_0, r_2, r_3)	{18}{0, 2, 12, 15}{9, 14, 19}{3, 11, 16}	9.35
(r_1, r_0, r_3, r_2)	{18}{0, 2, 12, 15}{9, 10, 14}{3, 11, 19}	9.41
(r_1, r_2, r_0, r_3)	{4}{0, 2, 12, 15}{9, 18, 19}{3, 11, 16}	9.44
(r_1, r_2, r_3, r_0)	{4}{0, 2, 12, 15}{9, 18, 19}{3, 11, 16}	9.44
(r_1, r_3, r_0, r_2)	{18}{0, 2, 12, 15}{9, 10, 14}{3, 11, 19}	9.41
(r_1, r_3, r_2, r_0)	**{4}{0, 2, 12, 15}{9, 14, 18}{3, 11, 19}**	**9.51**
(r_2, r_0, r_1, r_3)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_2, r_0, r_3, r_1)	{12}{2, 6, 7, 15}{0, 9, 18}{3, 11, 19}	9.04
(r_2, r_1, r_0, r_3)	{4}{2, 6, 12, 15}{0, 9, 18}{3, 11, 19}	9.25
(r_2, r_1, r_3, r_0)	{4}{2, 6, 12, 15}{0, 9, 18}{3, 11, 19}	9.25
(r_2, r_3, r_0, r_2)	{12}{0, 9, 18}{1, 10, 14}{3, 11, 19}	8.79
(r_2, r_3, r_2, r_0)	{4}{0, 9, 18}{10, 12, 14}{3, 11, 19}	8.98
(r_3, r_0, r_1, r_2)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_3, r_0, r_2, r_1)	{12}{2, 6, 7, 15}{9, 18, 19}{0, 3, 11}	8.98
(r_3, r_1, r_0, r_2)	{18}{2, 6, 12, 15}{9, 14, 19}{0, 3, 11}	9.10
(r_3, r_1, r_2, r_0)	{4}{2, 6, 12, 15}{9, 18, 19}{0, 3, 11}	9.19
(r_3, r_2, r_0, r_1)	{4}{2, 6, 7, 15}{9, 12, 18}{0, 3, 11}	8.91
(r_3, r_2, r_1, r_0)	{4}{2, 6, 7, 15}{9, 12, 18}{0, 3, 11}	8.91

Table 1: Comparisons among assignment strategies [76]

As it is showing in the Table 1, the optimum solution is (r_1, r_3, r_2, r_0) and the total is 9.51.

The solution of this problem is showing as Finger 4 and Figure 5.

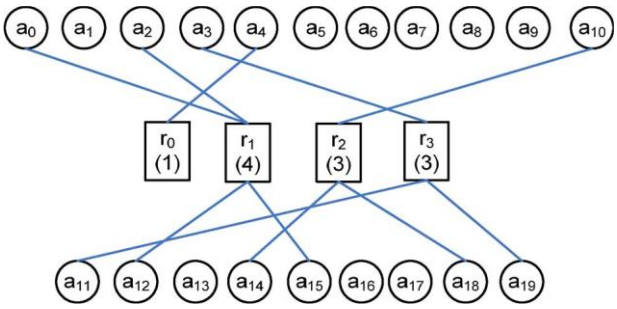

Figure 5: Solution of Figure 3 [76].

6.5 Classification Group Role Assignment Problem

The Group Role Assignment Problem can be classified into three types: Simple Role Assignment (SGRAP), Rated Role Assignment (RGRAP), and Weighted Role Assignment (WGRAP). Each one of these types has been defined and its objective function also has been found. In addition, Zhu et al [76] have clarified that the GRAP can transfer to a Generalized Assignment Problem by adjusting the number of agents and role. And then the problem can be solved by K-M algorithm. Since K-M algorithm is defined as a minimization algorithm for square matrices of GAPs created by Kuhn [33] Therefore, they apply this algorithm to GRAP and show that the result of solving this problem could not be a solution. For that, they provide some definitions and conditions to avoid the incorrect optimal result. They also prove the second type of the GRAP which is called RGRAP can be transformed into a Generalized Assignment Problem. They also provide an algorithm to solve the Rated Role Assignment. For the third type which is Weighted Role Assignment they prove it can be solved by the same algorithm that is used for the RGRAP. Consequently, they show the Simple Role Assignment is a particular

case of the RGRAP. The final part of their article [76] has included implementation, performance analysis and case study by simulation.

6.6 Conclusion

In this chapter, we have shown the Multiple Assignment Problem (MAP) is a generalized of the Assignment problem. We also introduced some particular situations of the MAP. One of these is called Group Role Assignment Problem (GRAP). In addition, in some cases, GRAP can be defined as a Multiple Knapsack Problem and indicates to the GAP. We also provide an example of GRAP which has been proposed and solved by Zhu et al [76].

Conclusion

We have presented in this research the Multiple Knapsack Problem which is a combinatorial optimization problem and known as an NP – hard problem. The MKP is a particular case of the Generalized Assignment Problem that is defined of minimizing cost or maximizing the profit of assigning tasks to only one agent while the sum of weight does not exceed the capacity. Contrary to the GAP the weight w_j of items j are independent to the Knapsack i. The $(0-1)$ Multiple Knapsack Problem is a generalized of the $(0-1)$ Knapsack Problem which treats the case of one knapsack.

The MKP and KP are usually solved by using the Branch and Bound and Dynamic Programming. However, for the MKP, Branch and Bound methods seem to behave better than Dynamic Programming. In this research, we had proposed a new method called Adapted Transportation Algorithm to solve the KP and its general form the MKP. It is based on linking the Knapsack Problem and the Multiple Knapsack Problem to the Linear Transportation Problem. Then an Adapted Transportation Algorithm is applied to find optimal solution. We can also notice that the Adapted Transportation Algorithm allows us to consider the situation where the profit unit for the items is dependent of the knapsack. Therefore, we have a matrix of profit $(P_{i,j})$ instead of a vector P_j.

For the future study, we will need to study the complexity of the ATA algorithm. We also need to implement it and test its robustness. That might allow us to compare it with the

other existing methods. This approach can also be extended to solve some of the KP such as the Subset-sum problem when weight is equal to profit, the bounded knapsack problem, the knapsack-like problems, and 2-dimensional knapsack problem [29]. It also can be extended to solve the multiple knapsack problem with assignment restrictions when each item can only assign to a specified subset of the knapsacks [29] and the multiple subset-sum problem.

References

1. Amini, M. M., & Racer, M. (1994). A rigorous computational comparison of alternative solution methods for the generalized assignment problem. *Management Science, 40*(7), 868-890.
2. Amini, M. M., & Racer, M. (1995). A hybrid heuristic for the generalized assignment problem. *European Journal of Operational Research, 87*(2), 343-348.
3. Asahiro, Y., Ishibashi, M., & Yamashita, M. (2003). Independentand cooperative parallel search methods for the generalized assignment problem. *Optimization methods and Software*, *18*(2), 129-141.
4. Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, *46*(3), 316-329.
5. Beale, E. M. L., & Tomlin, J. A. (1970). Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *OR*, *69*(447-454), 99.
6. Black, J. S. (1988). Work role transitions: A study of American expatriate managers in Japan. *Journal of International Business Studies*, *19*(2), 277-294.
7. Caprara, A., Kellerer, H., & Pferschy, U. (2000). The multiple subset sum problem. *SIAM Journal on Optimization*, *11*(2), 308-319.
8. Cattrysse, D. G., Salomon, M., & Van Wassenhove, L. N. (1994). A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, *72*(1), 167-174.

9. Ceselli, A., & Righini, G. (2006). A branch-and-price algorithm for the multilevel generalized assignment problem. *Operations research*, *54*(6), 1172-1184.
10. Chalmette L, Gelders L (1976) Lagrangian relaxation for a generalized assignment type problem. In: Advances in OR. EURO, North Holland, Amsterdam, pp 103–1.
11. Chu, P. C., & Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, *24*(1), 17-23.
12. Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations research*, *5*(2), 266-288.
13. de Farias Jr, I. R., Johnson, E. L., & Nemhauser, G. L. (2000). A generalized assignment problem with special ordered sets: a polyhedral approach. *Mathematical Programming*, *89*(1), 187-203.
14. Dıaz, J. A., & Fernández, E. (2001). A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research*, *132*(1), 22-38.
15. Drexl, A. (1991). Scheduling of project networks by job assignment. *Management Science*, *37*(12), 1590-1602.
16. Feltl H, Raidl GR (2004) An improved hybrid genetic algorithm for the generalized assignment problem. In: SAC ’04; Proceedings of the 2004 ACM symposium on Applied computing. ACM Press, New York, pp 990–995
17. Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management science*, *27*(1), 1-18.

18. Fisher, M. L., Jaikumar, R., & Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management Science*, *32*(9), 1095-1103.

19. Fukunaga, A. S., & Korf, R. E. (2007). Bin completion algorithms for multicontainer packing, knapsack, and covering problems. *Journal of Artificial Intelligence Research*, *28*, 393-429.

20. Gavish, B., & Pirkul, H. (1991). Algorithms for the multi-resource generalized assignment problem. *Management science*, *37*(6), 695-713.

21. Glover, F. (1968). Surrogate constraints. *Operations Research*, *16*(4), 741-749.

22. Glover, F., Hultz, J., & Klingman, D. (1978). Improved computer-based planning techniques, Part 1. *Interfaces*, *8*(4), 16-25.

23. Guignard, M., & Rosenwein, M. B. (1989). Technical Note—An Improved Dual Based Algorithm for the Generalized Assignment Problem. *Operations Research*, *37*(4), 658-663.

24. Haddadi S (1999) Lagrangian decomposition based heuris- tic for the generalized assignment problem. Inf Syst Oper Res 37:392–402

25. Haddadi, S., & Ouzia, H. (2004). Effective algorithm and heuristic for the generalized assignment problem. *European Journal of Operational Research*, *153*(1), 184-190.

26. Hung, M. S., & Fisk, J. C. (1978). An algorithm for 0- 1 multiple- knapsack problems. *Naval Research Logistics (NRL)*, *25*(3), 571-579.

27. Ingargiola, G., & Korsh, J. F. (1975). An algorithm for the solution of 0-1 loading problems. *Operations Research*, *23*(6), 1110-1119.

28. Jörnsten, K., & Näsberg, M. (1986). A new Lagrangian relaxation approach to the generalized assignment problem. *European Journal of Operational Research*, *27*(3), 313-323.
29. Kellerer, H., Pferschy, U., & Pisinger, D. Knapsack problems. 2004.
30. Klastorin, T. D. (1979). Note-On the Maximal Covering Location Problem and the Generalized Assignment Problem. *Management Science*, *25*(1), 107-112.
31. Kogan, K., & Shtub, A. (1997). DGAP-the dynamic generalized assignment problem. *Annals of Operations Research*, *69*, 227-239.
32. Kolesar, P. J. (1967). A branch and bound algorithm for the knapsack problem. Management science, 13(9), 723-735.
33. Kuhn, H. W. (2005). The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, *52*(1), 7-21.
34. Lin, B. M., Huang, Y. S., & Yu, H. K. (2001). On the variable-depth-search heuristic for the linear-cost generalized assignment problem. *International journal of computer mathematics*, *77*(4), 535-544.
35. Lorena, L. A. N., & Narciso, M. G. (1996). Relaxation heuristics for a generalized assignment problem. *European Journal of Operational Research*, *91*(3), 600-610.
36. Lorena, L. A., Narciso, M. G., & Beasley, J. E. (2002). A constructive genetic algorithm for the generalized assignment problem. *Evolutionary Optimization*, *5*, 1-19.
37. Lourenço HR, Serra D (1998) Adaptive approach heuristics for the generalized assignment problem. Technical Report 288, Department of Economics and Business, Universitat Pompeu Fabra, Barcelona

38. Lourenço HR, Serra D (2002) Adaptive search heuristics for the generalized assignment problem. Mathw Soft Comput 9(2–3):209–234

39. Manne, A. S. (1958). A target-assignment problem. *Operations Research*, *6*(3), 346-351.

40. Martello S, Toth P (1981) An algorithm for the general- ized assignment problem. In: Brans JP (ed) Operational Re- search ’81, 9th IFORS Conference, North-Holland, Amster- dam, pp 589–603

41. Martello, S., & Toth, P. (1980). Solution of the zero-one multiple knapsack problem. *European Journal of Operational Research*, *4*(4), 276-283.

42. Martello, S., & Toth, P. (1981). A bound and bound algorithm for the zero-one multiple knapsack problem. *Discrete Applied Mathematics*, *3*(4), 275-288.

43. Martello, S., & Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc..

44. Martello, S., & Toth, P. (1995). The bottleneck generalized assignment problem. *European journal of operational research*, *83*(3), 621-638.

45. Martello, S., Pisinger, D., & Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. Management Science, 45(3), 414-424.

46. Mazzola, J. B., & Neebe, A. W. (1988). Bottleneck generalized assignment problems. *Engineering Costs and Production Economics*, *14*(1), 61-65.

47. Monfared, M. A. S., & Etemadi, M. (2006). The impact of energy function structure on solving generalized assignment problem using Hopfield neural network. *European journal of operational research*, *168*(2), 645-654.

48. Najadat, F. A., Kanaan, G. G., Kanaan, R. K., Aldabbas, O. S., & Al-Shalabi, R. F. (2013). Genetic Algorithm Solution of the Knapsack Problem Used in Finding Full Issues in the Holy Quran Based on the Number (19). *Computer and Information Science*, *6*(2), 18.

49. Narciso, M. G., & Lorena, L. A. N. (1999). Lagrangean/surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, *114*(1), 165-177.

50. Nauss RM (2005) The elastic generalized assignment prob- lem. J Oper Res Soc 55:1333–1341

51. Nauss, R. M. (2003). Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS Journal on Computing*, *15*(3), 249-266.

52. Osman, I. H. (1995). Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Operations-Research-Spektrum*, *17*(4), 211-225.

53. Osorio, M. A., & Laguna, M. (2003). Logic cuts for multilevel generalized assignment problems. *European Journal of Operational Research*, *151*(1), 238-246.

54. Park, J. S., Lim, B. H., & Lee, Y. (1998). A Lagrangian dual-based branch-and-bound algorithm for the generalized multi-assignment problem. *Management Science*, *44*(12-part-2), S271-S282

55. Parker, R. G., & Rardin, R. L. (2014). *Discrete optimization*. Elsevier.

56. Pigatti, A., De Aragão, M. P., & Uchoa, E. (2005). Stabilized branch-and-cut-and-price for the generalized assignment problem. *Electronic Notes in Discrete Mathematics*, *19*, 389-395.

57. Pisinger, D. (1999). An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, *114*(3), 528-541.

58. Pisinger, D. (1999). Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, *33*(1), 1-14.

59. Romeijn, H. E., & Morales, D. R. (2000). A class of greedy algorithms for the generalized assignment problem. *Discrete Applied Mathematics*, *103*(1), 209-235.

60. Ronen, D. (1992). Allocation of trips to trucks operating from a single terminal. *Computers & operations research*, *19*(5), 445-451.

61. Ross, G. T., & Soland, R. M. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical programming*, *8*(1), 91-103

62. Ross, G. T., & Zoltners, A. A. (1979). Weighted assignment models and their application. *Management Science*, *25*(7), 683-696.

63. Samir, B., Yacine, L., & Mohamed, B. (2015). Local Search Heuristic for Multiple Knapsack Problem. *International Journal of Intelligent Information Systems*, *4*(2), 35-39.

64. Savelsbergh, M. (1997). A branch-and-price algorithm for the generalized assignment problem. *Operations research*, *45*(6), 831-841.

65. Singh, S. (2015). Note on Transportation Problem with new Method for Resolution of Degeneracy. *Universal Journal of Industrial and Business Management*, *3*(1), 26-36.

66. Ünal, A. N. (2013). A genetic algorithm for the multiple knapsack problem in dynamic environment. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 2).

67. Walkup, D. W., & MacLaren, M. D. (1964). *A multiple-assignment problem* (No. BOEING-MATH-347). BOEING SCIENTIFIC RESEARCH LABS SEATTLEWA.

68. Wilson JM (1997) A genetic algorithm for the generalised assignment problem. J Oper Res Soc 48:804–809

69. Yadav V. and S. Singh (2016), "Genetic Algorithms Based Approach to Solve 0-1 Knapsack Optimization Problem", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 4, Issue 5.

70. Yagiura, M., Ibaraki, T., & Glover, F. (2004). An ejection chain approach for the generalized assignment problem. *INFORMS Journal on Computing*, *16*(2), 133-151.

71. Yagiura, M., Ibaraki, T., & Glover, F. (2006). A path relinking approach with ejection chains for the generalized assignment problem. *European journal of operational research*, *169*(2), 548-569.

72. Yagiura, M., Yamaguchi, T., & Ibaraki, T. (1998). A variable depth search algorithm with branching search for the generalized assignment problem. *Optimization Methods and Software*, *10*(2), 419-441.

73. Yagiura, M., Yamaguchi, T., & Ibaraki, T. (1999). A variable depth search algorithm for the generalized assignment problem. In *Meta-heuristics* (pp. 459-471). Springer US.

74. Zhang, C. W., & Ong, H. L. (2007). An efficient solution to biobjective generalized assignment problem. *Advances in Engineering Software*, *38*(1), 50-58.

75. Zhu, H., & Zhou, M. (2006). Role-based collaboration and its kernel mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *36*(4), 578-589.

76. Zhu, H., Zhou, M., & Alkins, R. (2012). Group role assignment via a Kuhn–Munkres algorithm-based solution. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *42*(3), 739-750.