METAHEURISTIC APPROACHES FOR THE GENERALIZED ASSIGNMENT PROBLEM OF AN ONLINE EDUCATION WEBSITE

A Thesis

by

Merve Özer

Submitted to the

Graduate School of Sciences and Engineering

In Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in the

Department of Industrial Engineering

Özyeğin University

June 2021

METAHEURISTIC APPROACHES FOR THE GENERALIZED ASSIGNMENT PROBLEM OF AN ONLINE EDUCATION WEBSITE

Approved by:

Professor Ekrem Duman, Advisor,
Department of Industrial Engineering
Özyeğin University

Associate Professor Burcu Balçık,
Department of Industrial Engineering
Özyeğin University

Associate Professor Ali Fuat Alkaya
Department of Computer Engineering
Marmara University

Date Approved: 14 June 2021

To My Family

ABSTRACT

BinYaprak is a TurkishWIN (Turkish Women's International Network) initiative that offers role model stories for inspiration, educational content, real-life stories, and a networking platform through online and offline events. BinYaprak online education website aims to create ways for content and network aggregation between experts and learners for the ease of knowledge discovery. Experts are the people who join the platform for free in order to share their knowledge and experience with interested learners. Learners join the platform for free to learn new skills, access new networks, and find out new jobs and opportunities. Thus, this study aims to provide an assignment of a learner to an expert which satisfies both sides as much as possible. An expert can be assigned to more than one learner ensuring each learner is assigned at most one expert subject to each expert's capacity, thus the problem becomes a generalized assignment (GAP) which is known as an NP-Hard problem. In this study, we present a new implementation of a nature-inspired metaheuristic algorithm called Migrating Birds Optimization (MBO) and a hybrid of Simulated Annealing (SA) and Tabu Search (TS) methods in order to solve the GAP of BinYaprak online education website. In our computational study, we tested both MBO and hybrid SA/TS on small and large-sized instances of GAP of BinYaprak website. Also, we used Gurobi Python API as a baseline reference against which we can compare our heuristic methods' performances. Our numerical analyzes show that the hybrid SA/TS outperforms the MBO in terms of solution quality and computational effort, hence hybrid SA/TS can be used in practice to obtain high-quality solutions.

Keywords: Generalized Assignment Problem, Combinatorial Optimization, Metaheuristics, Migrating Birds Optimization, Hybrid Simulated Annealing and Tabu Search.

ÖZETÇE

BinYaprak, çevrimiçi eğitim içerikleri ve ilham veren başarı hikayeleri sunmanın yanı sıra çevrimiçi ve çevrimdışı etkinlikler aracılığıyla bir ağ platformu sunan TurkishWIN ’in (Türk Kadınlarının Uluslararası Ağı) bir girişimidir. BinYaprak çevrimiçi eğitim web sitesi, bilgi keşfinin kolaylığı için uzmanlar ve öğreniciler arasında içerik ve ağ (networking) oluşturmayı amaçlamaktadır. Uzmanlar, bilgi ve deneyimlerini ilgili öğrenicilerle paylaşmak için platforma ücretsiz olarak katılan kişilerdir. Öğreniciler ise yeni beceriler öğrenmek, yeni ağlar kurmak, yeni işler ve fırsatlar hakkında bilgi edinmek için platforma ücretsiz olarak katılırlar. Bu nedenle bu çalışma, BinYaprak online eğitim websitesinin kullanıcılarını mümkün olduğunca tatmin etmeye çalışan, bir uzmana öğrenici ataması sağlamayı amaçlamaktadır. Bir uzmana birden fazla öğrenici atanabilir ve her bir öğrenici, her bir uzmanın ayırabileceği toplam seans sayısına bağlı olarak, en fazla bir uzmana atanabilir, böylece NP-zor bir problem olan genelleştirilmiş atama problemi (GAP) oluşur. Bu çalışmada, BinYaprak çevrimiçi eğitim web sitesinin genelleştirilmiş atama problemi (GAP)’ ini çözmek için, doğadan ilham alan yeni bir metasezgisel olan Göç Eden Kuşlar Optimizasyon algoritması ve bir hibrit Benzetilmiş Tavlama ve Tabu Arama algoritmasının uygulamasını sunmaktayız. Göçmen Kuşlar Optimizasyon ve hibrit Benzetilmiş Tavlama ve Tabu Arama algoritmaları, BinYaprak web sitesinin küçük ve büyük boyutlu test problemleri kullanılarak test edilmiştir. Ayrıca, Gurobi Python API’ ını metasezgisel yöntemlerimizin performanslarını karşılaştırabileceğimiz temel bir referans olarak kullandık. Sayısal analizlerimiz, hibrit Benzetilmiş Tavlama ve Tabu Arama algoritmasının, çözüm kalitesi ve hesaplama süresi bakımından, Göçmen Kuşlar Optimizasyon algoritmasından daha iyi performansa sahip olduğunu göstermektedir. Dolayısıyla, tasarladığımız hibrit Benzetilmiş Tavlama ve

Tabu Arama algoritması, BinYaprak online eğitim websitesinin genelleştirilmiş atama problemine iyi sonuçlar sağlamak için kullanılabilir.

Anahtar Kelimeler: Genelleştirilmiş Atama Problemi, Kombinatoryal Optimizasyon, Metasezgiseller, Göçmen Kuşlar Optimizasyon Algoritması, Hibrit Benzetilmiş Tavlama ve Tabu Arama Algoritması.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and appreciation to my advisor Professor Ekrem Duman for suggesting the topic of this thesis and for his expert guidance throughout my study and research. He provided me with the unique opportunity to work in the research area of metaheuristics. It was a great honor to work under his guidance.

I would like to express my special thanks to Assoc. Prof. Nagihan Çömez Dolgan for her constant guidance, encouragement, support and belief in me since my undergraduate years.

I would also like to thank my family for their life-long love, faith and support for me during this year.

Finally, I would like to thank all the professors in the Industrial Engineering department at Özyeğin University for the support I have received during this year.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

There are many applications of assignment problems that often appear in real life such as assigning origins to destinations, workers to the jobs, resources to a variety of activities, students to projects, facilities to locations, etc. To begin with, we encounter a one-to-one matching of *m* agents and *m* tasks in the classical assignment problem to obtain the minimum assignment cost. We can achieve an optimal solution for the classical assignment problem in polynomial time. One of the quintessential examples of the classical assignment problem is assigning jobs to workers to find the minimum worker-job assignment cost. One of the most popular algorithms that solves the classical assignment problem in polynomial time is the Hungarian Algorithm, which is introduced by Khun (1955).

The knapsack problem may be encountered in the context of resource allocation with financial constraints. The 0-1knapsack problem offers a significant model for financial planning in project selection, portfolio management, and cargo loading. In the context of project selection, for instance, a firm needs to choose a subset of projects that maximizes return based on the budget constraint. The 0/1 constraint in the problem is that if a project is selected entirely $x_i=1$ or not selected at all, $x_i= 0$.

Furthermore, when we need to assign a set of facilities to several locations to find the minimum transportation cost between the facilities, the Quadratic Assignment Problem arises, which is classified as the NP-Hard combinatorial optimization problem. In the QAP, the number of locations and the facilities is usually equal to each other. Since the QAP is NP-hard, there are several heuristic techniques developed to solve the QAP, one of the popular heuristic techniques is the greedy randomized adaptive search procedures

(GRASPs) proposed by Feo and Resende (1995) to solve difficult combinatorial optimization problems.

Generalized Assignment Problem (GAP) is another popular combinatorial optimization problem, which considers the allocation of limited resources to a variety of activities considering the capacity of each resource to minimize the total assignment cost. For instance, the assignment of customer demands to production facilities while minimizing the total customer assignment cost is one of the real-life applications of GAP (Geunes, 2014). Since the GAP is an NP-Hard problem, it is intractable to solve the large instances in polynomial time, thus there are several heuristic and metaheuristic methods are introduced (Öncan, 2007). For instance, Martello and Toth (1981) developed a heuristic algorithm (MTH) to solve GAP. Moreover, Baykasoğlu et al. (2007) present an application of the Artificial Bee Colony Algorithm to solve GAP, and noted that ABC is very effective in solving small and medium-sized GAPs. Diaz and Fernandez (2001) applied Tabu Search with long-term memory functions on GAP and obtained very high-quality solutions when compared to other methods such as Simulated Annealing and Genetic Algorithm. In this study, we designed a hybrid Simulated Annealing and Tabu Search with a diversification strategy and a Migrating Birds Optimization (MBO) algorithm to solve the GAP of an online education website. To the best of our knowledge, this paper is the first study on applying MBO to a real-life application of GAP.

2. THE BINYAPRAK PROBLEM

BinYaprak is a TurkishWIN (Turkish Women's International Network) initiative that offers role model stories for inspiration, educational content, and real-life stories as well as a networking platform through online and offline events. BinYaprak online education website aims to create ways for content and network aggregation between experts and learners for the ease of knowledge discovery. Experts are the people who join the platform for free in order to share their knowledge and experiences with interested learners. Learners join the platform for free so as to learn new skills, access new networks, and find out about new jobs and opportunities. Therefore, this study aims to provide an assignment of a learner to an expert which satisfies both sides as much as possible. Each expert specifies a number of sessions that he or she can allocate for the potential learners each week, thus the number of sessions becomes the expert capacity. If a number of tasks need to be assigned to a number of agents in such a way that allows the assignment of multiple tasks to an agent based on the capacity restriction on the agents, then the generalized assignment problem arises (Kundakcioglu & Alizamir, 2008). The generalized assignment problem becomes different from the classical assignment problem (GAP) such that an agent consumes not only one but a number of resources while performing the tasks that are assigned to him. In our context, an expert may be assigned to multiple learners ensuring each learner is assigned at most one expert not exceeding the capacity of the expert is called a feasible solution, hence, the problem becomes a generalized assignment problem (GAP) which is known to be NP-Hard (Kundakcioglu & Alizamir, 2008).

2.1 Mathematical Model

We consider *i* learners L= {1,2,3,4,..,i}, indexed by *l* , and *j* experts E={1,2,3,4,..,j}, indexed by *e*. Each expert and learner indicate a list of sectors, expertise and experience that they are interested in. We wish to find a maximum score assignment which is based on the constraints of assigning each learner to only one expert while aiming to ensure that both sides are as happy with the assignment as possible. As the number of experts and learners does not have to be equal, an assignment does not guarantee that each learner will be assigned. Each expert *e* has an available capacity of C_e indicating the total number of sessions that an expert allocates each week. Duration of each session is determined by the expert, and a_{el} is the capacity (number of sessions) used when learner *l* is assigned to expert *e*. The decision variable, x_{el}, will thus compose of a binary assignment variable. If learner *l* is assigned to expert *e*, x_{el} = 1 and 0 otherwise, and s_{el} is the score of assigning a learner *l* to expert *e* which is obtained through the score matrix algorithm that we designed. GAP of Binyaprak Online Education website can be formulated such that:

$$Maximize \qquad \sum_{l \in L} \sum_{e \in E} s_{el} x_{el}$$

$$Subject\ to:$$

$$\sum_{l \in L} a_{el} x_{el} \leq C_e \qquad e \in E, \tag{1}$$

$$\sum_{e \in E} x_{el} \leq 1, \qquad l \in L, \tag{2}$$

$$x_{e,l} \in \{0,1\}, \qquad e \in E,\ l \in L. \tag{3}$$

The objective function maximizes the total assignment score, and the first condition (1), which is called expert capacity, signifies that we do not violate the capacity of an expert. The second condition (2) ensures that each learner $l \in L$ can be assigned to at most one available expert, which is called the assignment constraint.

2.2 Creating an Online Education Website for the Binyaprak

We firstly developed a website for BinYaprak and the snapshot of the website is shown below. Anyone can register on the website and join special interest groups such as Book Club, Data Science Club and Engineers Club, etc., where one can build new networks. What is more, there are several blog posts which are about; for instance, personal development, data science, career and education, etc. available in the "Writings" part as well as there are many educational videos about the choice of profession, healthy life, university life, internships, etc., in the "Videos" part. All these contents are available for free by just registering on the website. The main purpose of the BinYaprak online education website is to create ways for content and network aggregation between experts and learners for the ease of knowledge discovery. We define someone as an *expert* if she or he wants to share information about a sector, a profession, or an experience that he or she is interested in, on the other hand, a *learner* is a person who wants to learn about a specific sector, profession and an experience. Thus, in order for experts to share information, they have to register on the website and create an account at first. Then, we collect some basic personal information, and we require experts to fill out a questionnaire which is about their interested sectors, professions/expertise and experiences in the "My Contributions" part. Learners also need to fill out a questionnaire, in the "My Interests" part, about what sectors, professions/expertise and experiences they are interested in and

want to get information. Therefore, we need to design an algorithm to provide with the BinYaprak website an assignment of each learner to an expert based on the preference list each week.

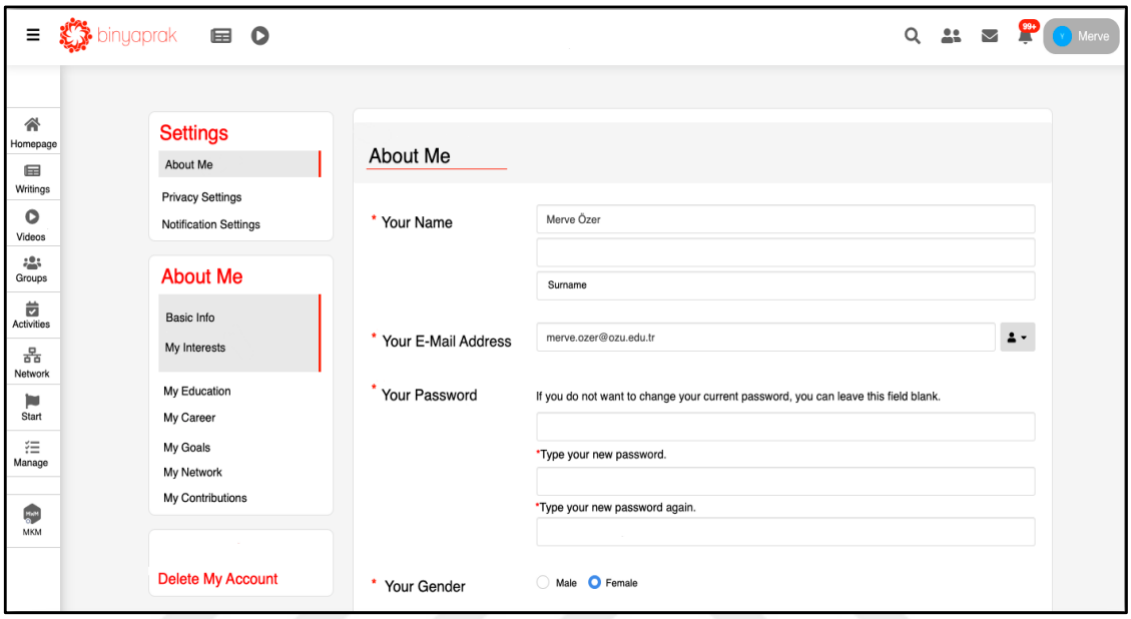

Figure 1: The snapshot of the BinYaprak Website.

2.3 Data Collection

We used a real data-set collected based on real preferences from experts and learners registered at BinYaprak online education website. The data gathered through an online survey that we created on the BinYaprak website. Both experts and learners have to fill out questionnaires about sector, expertise and experience preferences while registering on the website. Figure 2 below shows the survey that learners fill out. Each learner is allowed to choose up to three sectors, expertise and experiences that they are interested in.

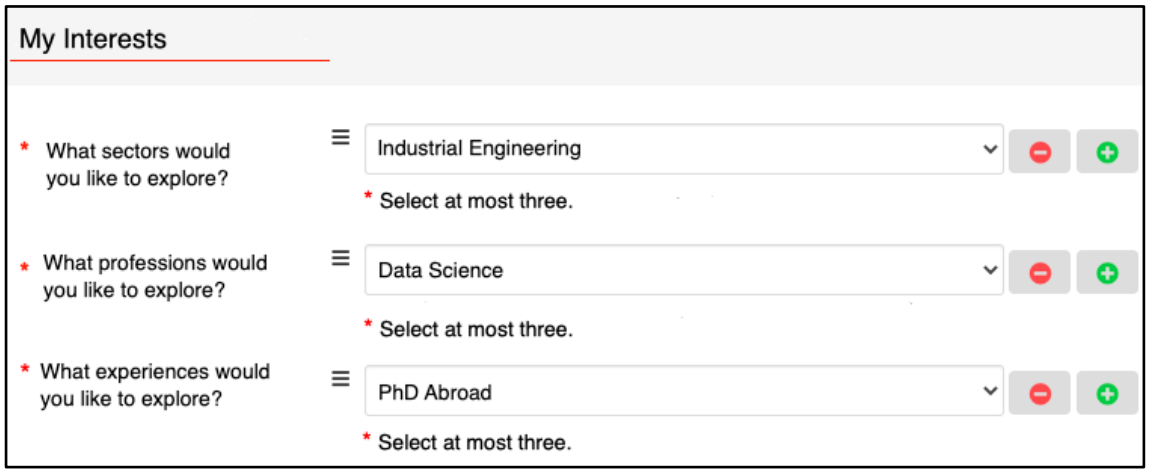

Figure 2: Learners' Preference List

Figure 3 below shows the survey that experts fill out. Each expert is allowed to specify up to three sector, expertise and experience that they want to share information about.

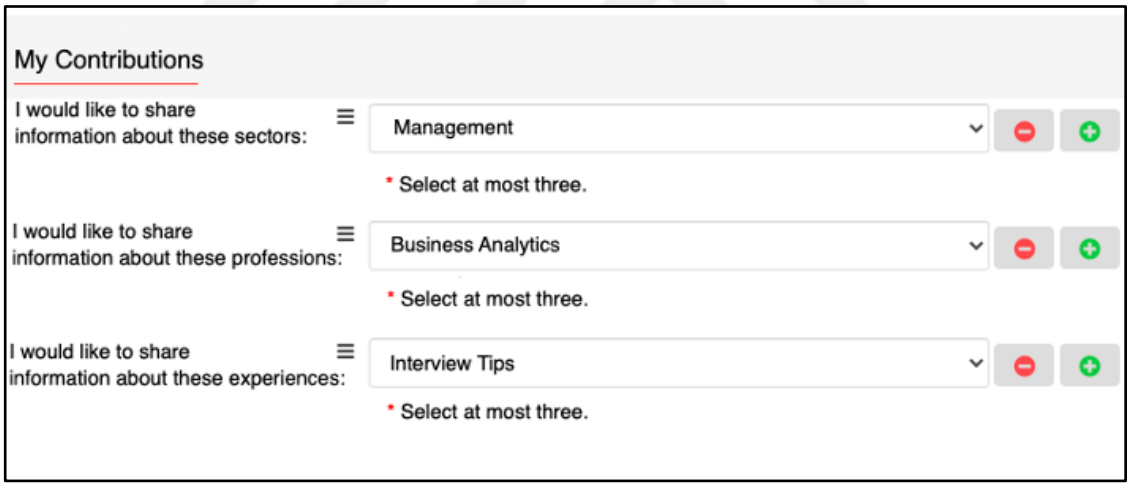

Figure 3: Experts' Preference List

2.4 The Score Matrix Algorithm

The steps of the score matrix algorithm that we designed to calculate the assignment score s_{el} are as follows:

Step 1. Each expert and learner indicate a list of sector, expertise and experience, which will be ranked in order of preference.

Step 2. A predefined score is assigned to each sector, expertise and experience based on its rank. The first ranked preference in each category will get the highest score and the last ranked preference will get the least score.

Step 3. The scores of experts and learners who preferred the same sector, expertise and experience, which is based on its rank, are summed up, and thus creating the assignment score.

After performing **Step 1** and **Step 2**, the following example tables are obtained in which the notation *E* indicates the expert, *L* indicates the *learner*. Table 1 provides the preference list of each expert and associated score of each preference, likewise, Table 2 provides the preference list and associated scores of each preference of learners who want to get information about the specified sector, expertise, and experience.

Table 1. Preference list of each expert.

	Sector			Expertise			Experience		
Rank	1	2	3	1	2	3	1	2	3
Score	18	12	6	18	12	6	18	12	6
E1	Computer/Electronics	Information Technologies	Education	Data Science	Software	-	Internship	Choice of Profession	-
E2	Pharmacy	Medicine and Wellness	Biology	-	-	-	Cash Management	Choice of Profession	-
E3	Consultancy	Financial Services	-	Resource Management	-	-	Global Career		
E4	Education	Publishing		E-Commerce	Social Media	-	Erasmus Education	Erasmus Internship	-
E5	Financial Services	-	-	Business Analytics	-	-	Cash Management	Stress Management	Interview Tips
E6	Law	-	-	Consultancy	-	-	Job Search	Interview Tips	-
E7	General Public	Corporate Service	Insurance Business	Product Management	-	-	Choice of Profession	Leadership	-
E8	Medical	-	-	-	-	-	Entrepreneurship	Interview Tips	-
E9	Fine Arts	Design	-	Design	Media		Networking	Choice of Profession	-
E10	Software Development	Hardware and Computer Networks	-	Artificial Intelligence	Code Quality	Social Media	Interview Tips	Job Search	-

Table 2**.** Preference list of each learner.

	Sector			Expertise			Experience		
Rank	1	2	3	1	2	3	1	2	3
Score	9	6	3	9	6	3	9	6	3
L1	Information Technologies	Hardware and Computer Networks	Software Development	E-Commerce	Social Media	-	Internship	Choice of Profession	Interview Tips
L2	Law	-	-	Media	Consultancy	-	Job search	Interview Tips	-
L3	Tourism	Consultancy		Consultancy	-	-	Global Career	Choice of Profession	-
L4	Manufacturing	-		Marketing	Communication	Social Media	Job search	Internship	-
L5	Fine Arts	Design		Media	-	-	Job search	Internship	-
L6	Entertainment	Leisure and Travel		Media	-	-	Erasmus	Graduate Study	-
L7	Real Estate	-		Marketing	-	-	Cash Management	-	-
L8	Computer/Electronics	Consultancy		Consultancy	-	-	Cash Management	-	-
L9	Energy/Mining	-		Resource Management	Manufacturing	-	Entrepreneurship	-	-
L10	Fine Arts	Design		Communication	Media	-	Choice of Profession	-	-

After performing **Step 3.** the algorithm provides the score matrix which demonstrates the assignment score of each expert-learner pair below. The last column represents the capacity of each expert which is the total number of sessions that an expert allocates for learners each week.

Table 3. The Score Matrix

E/L	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	C
E1	66	0	0	0	0	0	0	27	0	0	4
E2	18	0	0	0	0	0	90	27	0	0	3
E3	0	0	51	0	0	0	0	24	27	0	2
E4	27	0	0	0	0	0	0	0	0	0	3
E5	9	12	0	0	0	0	27	27	0	0	1
E6	15	96	27	27	27	0	0	27	0	0	1
E7	24	0	0	0	0	0	0	0	0	0	2
E8	15	18	0	0	0	0	0	0	27	0	3
E9	18	21	0	0	66	21	0	0	0	45	3
E10	60	45	0	21	21	0	0	0	0	0	2

3. ASSIGNMENT PROBLEMS

In this section, different assignment problems in the literature, their mathematical models and approaches to solve these problems will be explained.

3.1 The Classical Assignment Problem

The purpose of the classical assignment problem is to achieve minimum cost assignment of a number of items *i* to a group of resources *j*. In the classical assignment problem, each resource is allocated one item and each item is assigned to exactly one resource, where the allocation of a given item to a specific resource is associated with a cost. There is a one-to-one matching of agents and tasks in the classical assignment problem, which is solved in polynomial time (Kundakcioglu & Alizamir, 2008). The mathematical model of the classical assignment problem is shown below:

$$Minimize \qquad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$Subject\ to:$$

$$\sum_{i \in I} x_{ij} = 1 \qquad j \in J, \qquad (1)$$

$$\sum_{j \in J} x_{ij} = 1, \qquad i \in I, \qquad (2)$$

$$x_{i,j} \in \{0, 1\}, \qquad i \in I,\ j \in J \qquad (3)$$

where I represent the index set of items, J represents the index set of resources j, and c_{ij} represents the cost of assigning item i to resource j. Also, the decision variable x_{ij} is defined as follows: $x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is assigned to resource j,} \\ 0 & \text{otherwise.} \end{cases}$

3.1.1 The Hungarian Algorithm

We can find a solution to the classical assignment problem in polynomial time by a best-known method named as Hungarian Algorithm due to its special structure. The application of the Hungarian Algorithm to solve the classical assignment problem is illustrated below.

Example 1

Assume there are 6 items requires to be assigned to 6 resources, and for each item and resource, there is a cost for assigning an item to one resource. Each item needs to be assigned to at most one resource, with no two resources are assigned the same items while minimizing the total cost. The costs of assigning an item to a resource are given in the following table.

Table 4. Assignment cost of each item to a resource.

	0	**1**	**2**	**3**	**4**	**5**
0	90	80	75	70	60	85
1	90	85	55	65	95	60
2	125	95	90	60	105	60
3	100	110	95	115	95	140
4	50	100	90	95	60	100
5	110	80	95	75	90	90

The implementation of the Hungarian method on Example 1 is as follows.

Step 1. Row Subtraction

The minimum element of each row is deducted from all the elements of that row, the following matrix is obtained:

	0	**1**	**2**	**3**	**4**	**5**
0	30	20	15	10	0	25
1	35	30	0	10	40	5
2	65	35	30	0	45	0
3	5	15	0	20	0	45
4	0	50	40	45	10	50
5	35	5	20	0	15	15

Step 2. Column Subtraction

The minimum element of each column is deducted from all the elements of that column, the following matrix is obtained, please note that if there is a zero in each row, then the column subtraction step can be skipped.

	0	**1**	**2**	**3**	**4**	**5**
0	30	15	15	10	0	25
1	35	25	0	10	40	5
2	65	30	30	0	45	0
3	5	10	0	20	0	45
4	0	45	40	45	10	50
5	35	0	20	0	15	15

Step 3. Make assignments in the opportunity cost table and cover all zeros by drawing a minimum number of lines.

	0	**1**	**2**	**3**	**4**	**5**
0	30	15	15	10	**[0]**	25
1	35	25	**[0]**	10	40	5
2	65	30	30	**[0]**	45	0
3	5	10	0	20	0	45
4	**[0]**	45	50	45	10	50
5	35	**[0]**	20	0	15	15

Step 4. The number of assignments = 5, number of rows = 6, which is not equal, so the solution is not optimal, thus we need to increase the number of zeros by subtracting a minimum element, which is 5, from all the elements of each row.

	0	**1**	**2**	**3**	**4**	**5**
0	25	10	15	5	0	20
1	30	20	0	5	40	0
2	65	30	35	0	50	0
3	0	5	0	15	0	40
4	0	45	45	45	15	50
5	35	0	0	0	20	15

Step 5. Optimal Assignment

The following zero in each row covers an optimal solution for the problem.

	0	**1**	**2**	**3**	**4**	**5**
0	25	10	15	5	**[0]**	20
1	30	20	0	5	40	**[0]**
2	65	30	35	**[0]**	50	0
3	0	5	**[0]**	15	0	40
4	**[0]**	45	45	45	15	50
5	35	**[0]**	0	0	20	15

The optimal solution is:

Item	Resource	Cost
0	4	60
1	5	60
2	3	60
3	2	95
4	0	50
5	1	80
Total Cost		405

3.2 The Generalized Assignment Problem (GAP)

The generalized assignment problem (GAP) is an example of the combinatorial optimization problems, which is an NP-Hard in the strong sense, since we are not likely to provide an exact solution for the problem in polynomial time. The assignment of limited resources to various activities required in both production and distribution of a good and service constitutes a majority of challenging operations planning problems (Geunes, 2014). Given a set of items and resources, we need to obtain a minimum cost assignment in which each available resource has an associated capacity limit. Allocation of customers to warehouses, storage space allocation, vehicle routing, facility location, knapsack, flexible manufacturing systems, warehouse allocation problems are examples of practical applications to GAP. The generalized assignment problem differs from the classical assignment problem in such a way that a number of items are required to be assigned to several resources while permitting the assignment of multiple items to a resource considering each resource's capacity. However, in the classical assignment, we encounter a one-to-one assignment. Consider *n* items I= {1,2,3,4,..,n}, indexed by i, and *m* resources J={1,2,3,4,..,m}, indexed by j. Resource j has an available capacity of K_j. Assigning item $i \in I$ to resource $j \in J$ consumes b_{ij} units of resource j capacity and results in a cost of c_{ij}. We need to minimize the total assignment cost based on the constraints of assigning each item to only one resource. Thus, generalized assignment problem can be modelled such that:

$$Minimize \qquad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$Subject\ to:$$

$$\sum_{i \in I} b_{ij} x_{ij} \leq K_j \qquad j \in J, \tag{1}$$

$$\sum_{j \in J} x_{ij} = 1, \qquad i \in I, \tag{2}$$

$$x_{ij} \in \{0, 1\}, \qquad i \in I, \ j \in J. \tag{3}$$

The objective function minimizes total assignment cost, and the first condition (1), which is called resource capacity, and means that the capacity of any resource is not violated. The second condition (2) ensures that each item i∈ I is assigned to only one available resource, which is called the assignment constraint. The decision variable x_{ij} is composed of a binary assignment, which means, $x_{ij} = 1$, If item *i* is assigned to resource *j*, and 0 otherwise.

Example 2. Resource Assignment Problem

A small instance of the GAP in which there are two resources *(j)* and seven items *(i)* is given below. The capacity of the **Resource 1** is 5 units while the capacity of **Resource 2** is12 units. The following Table 5. provides the assignment costs (c_{ij}) and the capacity consumption values (b_{ij}) respectively when item *i* is assigned to resource *j*.

Table 5. Assignment costs and capacity consumption values.

c_{ij} values							
Item, i	1	2	3	4	5	6	7
Resource 1	2	1	9	3	2	1	3
Resource 2	3	4	2	4	5	5	4

b_{ij} values							
Item, i	1	2	3	4	5	6	7
Resource 1	3	3	2	4	2	3	2
Resource 2	6	6	6	1	2	1	1

Source: Geunes, J. (2014). Operations Planning: Mixed Integer Optimization Models (Vol. 11). CRC Press.

We need to assign each item to exactly one resource based on units of capacity available while minimizing the total cost. An optimal solution for the small instance of GAP can be obtained by using Gurobi Python API.

Table 6. Gurobi output for the problem.

```
Optimize a model with 9 rows, 14 columns and 28 nonzeros
Model fingerprint: 0xa1f0c0c8
Variable types: 0 continuous, 14 integer (14 binary)
Coefficient statistics:
  Matrix range     [1e+00, 6e+00]
  Objective range  [1e+00, 9e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+01]
Found heuristic solution: objective 31.0000000
Presolve removed 9 rows and 14 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 4 available processors)

Solution count 1: 31

Optimal solution found (tolerance 1.00e-04)
Best objective 3.100000000000e+01, best bound 3.100000000000e+01, gap 0.0000%
```

Table 7. Optimal assignment for the problem.

Item	**Resource**	**Cost**
2	1	1
3	1	9
1	2	3
4	2	4
5	2	5
6	2	5
7	2	4
Total Cost		**31**

3.2.1 Approximation Algorithms

Approximation algorithms are efficient algorithms that provide a solution to combinatorial optimization problems which is provably near to optimal solution. Approximation algorithms are usually used when it is intractable to achieve an optimal solution as well as in some cases, they are used to provide a near-optimal solution quickly when an exact solution is not needed.

3.2.2 Linear Programming Relaxation

LP relaxation is one of the standard approaches in designing approximation algorithms to solve NP-Hard problems. Since the GAP is NP-Hard, we can use an LP

relaxation technique to obtain a near-optimal solution. We can relax the assignment constraint for Example 2. The assignment constraint which is $x_{ij} = 0$ or 1 is replaced by appropriate continuous constraints such as $x_{ij} \geq 0$ and $x_{ij} \leq 1$. We can apply LP relaxation on Example 2 and obtain the following solution with an objective function value of 27.17. Table 8. presents the following variable values:

Table 8. LP relaxation solution.

```
assign[Resource1,item1]: 0.3333333333333335
assign[Resource1,item2]: 1.0
assign[Resource1,item3]: 0.49999999999999983
assign[Resource2,item1]: 0.6666666666666665
assign[Resource2,item3]: 0.5000000000000002
assign[Resource2,item4]: 1.0
assign[Resource2,item5]: 1.0
assign[Resource2,item6]: 1.0
assign[Resource2,item7]: 1.0
Total cost score: 27.166666666666664
```

As we can see from the output that the LP relaxation solution is 27.167. We obtained the optimal solution value of 31 with Gurobi. One of the important concepts in this application is the **integrality gap** which is calculated as follows:

$$100\ x \frac{31 - 27.167}{31} \approx 12.4\%$$

We can also calculate the approximation ratio which is the ratio between the result obtained with the algorithm and the optimal cost or profit. For minimization problems, as in Example 2, we can calculate the approximation ratio as follows:

$$\frac{31}{27.167} = 1.141$$

For maximizations problems, the fraction is reversed.

3.2.3 A Heuristic Approach for the Generalized Assignment Problem

Martello and Toth (MTH) designed a heuristic algorithm to solve Generalized Assignment Problem, which is described as follows. Let f_{ij} be a measure of the desirability of assigning item i to resource j. MTH iteratively regards all the unassigned items, and then determines the item i^* which has the minimum difference between the smallest and the second smallest f_{ij}. The item i^* is then assigned to the resource for which f_{ij} is a minimum. In the second phase of the algorithm, the current solution is tried to be improved through a local exchange procedure. Four different f_{ij} functions are used (Toth & Martello, 1981). These are as follows:

a) $f_{ij} = c_{ij}$;

b) $f_{ij} = \frac{c_{ij}}{b_{ij}}$;

c) $f_{ij} = b_{ij}$;

d) $f_{ij} = \frac{b_{ij}}{K_j}$;

Let us consider execution MTH with $f_{ij}=b_{ij}$ for the Example 2.

n=7;

m=2;

$(c_{ij}) = \begin{pmatrix} 2 & 1 & 9 & 3 & 2 & 1 & 3 \\ 3 & 4 & 2 & 4 & 5 & 5 & 4 \end{pmatrix}$;

$(b_{ij}) = \begin{pmatrix} 3 & 3 & 2 & 4 & 2 & 3 & 2 \\ 6 & 6 & 6 & 1 & 2 & 1 & 1 \end{pmatrix}$; $(K_j) = \begin{pmatrix} 5 \\ 12 \end{pmatrix}$.

The first phase of the algorithm gives;

Iteration 1. $i^* = 3$: $d^* = -4$, $y_3 = 1$, $\overline{K_1} = 3$; cost of assignment: 9

Iteration 2. $i^* = 1$: $d^* = -3$, $y_1 = 1$, $\overline{K_1} = 0$; cost of assignment: 2

Iteration 3. $i^* = 2$: $d^* = -3$, $y_2 = 2$, $\overline{K_2} = 6$; cost of assignment: 4

Iteration 4. $i^* = 4$: $d^* = -3$, $y_4 = 2$, $\overline{K_2} = 5$; cost of assignment: 4

Iteration 5. $i^* = 6$: $d^* = -2$, $y_6 = 2$, $\overline{K_2} = 4$; cost of assignment: 5

Iteration 6. $i^* = 7$: $d^* = -1$, $y_7 = 2$, $\overline{K_2} = 3$; cost of assignment: 4

Iteration 7. $i^* = 5$: $d^* = -0$, $y_5 = 2$, $\overline{K_2} = 1$; cost of assignment: 5

Now, store the feasible solution z^h in y_i [resource to which item *i* is assigned]. Hence the total cost of assignment $z^h = 33$, and $y_i = [1\ 2\ 1\ 2\ 2\ 2\ 2]$, $(\overline{K}) = [0\ 1]$.

The second phase of the algorithm performs the exchanges:

Exchange item 1 and item 2;

$i = 1 : y_1 = 2$

$i = 2 : y_2 = 1$

and the optimal solution found is:

$z^h =$ **31**, and $y_i = [2\ 1\ 1\ 2\ 2\ 2\ 2]$, $(\overline{K}) = [0\ 1]$. As we can see, we improved the initial solution and obtained the optimal solution for the problem through MTH algorithm.

3.3 The Generalized Assignment for the Maximization Problems

The maximization version of the generalized assignment problem can be found in the literature in such a way that p_{ij} is defined as the profit of item i if it is assigned to the resource j. We need to assign each item to only one resource in order to maximize the total profit considering the capacity of each resource. The GAP for the maximization problems is formulated such that:

$$Maximize \qquad \sum_{i \in I} \sum_{j \in J} p_{ij} x_{ij}$$

$$Subject\ to:$$

$$\sum_{i \in I} b_{ij} x_{ij} \leq K_j \qquad j \in J, \tag{1}$$

$$\sum_{j \in J} x_{ij} = 1, \qquad i \in I, \tag{2}$$

$$x_{ij} \in \{0, 1\}, \qquad i \in I, \ j \in J. \tag{3}$$

3.3.1 A Heuristic Approach for Maximization Problems

Martello and Toth (1981) developed a heuristic to MAXGAP. We already defined f_{ij} which is the measure of the desirability of assigning item *i* to resource *j;* however, we need to do some adjustments for the f_{ij} function for the maximization problems. The $f_{ij} = b_{ij}$ function used for minimization problems becomes $f_{ij} = -b_{ij}$ for the maximization problems. Four different f_{ij} functions can be used for the maximization problems:

a) $f_{ij} = p_{ij}$ (the improvement phase can be skipped with this choice);

b) $f_{ij} = \frac{p_{ij}}{b_{ij'}}$

c) $f_{ij} = -b_{ij}$;

d) $f_{ij} = -\frac{b_{ij}}{K_j}$;

We iteratively consider all the items that are not assigned, and we need to find an item i^* which has the maximum difference between the largest and the second largest f_{ij}, and i^* is then assigned to the resource for which f_{ij^*} is a maximum. In the second phase of the algorithm, the current solution is improved through local exchange procedures.

Example 3.

Consider the following instance of generalized assignment problem. Given 8 items and 3 resources, we need to assign each item to only one resource based on units of capacity available with the aim of maximizing the total profit.

n=8

m=3

$$p_{ij} = \begin{pmatrix} 28 & 13 & 13 & 17 & 25 & 32 & 42 & 14 \\ 15 & 6 & 38 & 10 & 37 & 26 & 2 & 35 \\ 35 & 35 & 21 & 10 & 20 & 20 & 4 & 35 \end{pmatrix};$$

$$b_{ij} = \begin{pmatrix} 22 & 14 & 10 & 6 & 8 & 16 & 6 & 25 \\ 21 & 9 & 19 & 26 & 7 & 7 & 10 & 7 \\ 17 & 17 & 19 & 25 & 12 & 12 & 17 & 19 \end{pmatrix}; \quad (K_j) = \begin{pmatrix} 27 \\ 26 \\ 35 \end{pmatrix}$$

We can obtain the optimal solution for Example 3. by using Gurobi Python API. The optimal solution is 240.

Table 9. Gurobi output for the GAP.

```
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (mac64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 11 rows, 24 columns and 48 nonzeros
Model fingerprint: 0x98c60548
Variable types: 0 continuous, 24 integer (24 binary)
Coefficient statistics:
  Matrix range     [1e+00, 3e+01]
  Objective range  [2e+00, 4e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+01]
Found heuristic solution: objective 153.0000000
Presolve time: 0.00s
Presolved: 11 rows, 24 columns, 48 nonzeros
Variable types: 0 continuous, 24 integer (24 binary)

Root relaxation: objective 2.547692e+02, 8 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  254.76923    0    4  153.00000  254.76923  66.5%     -    0s
H    0     0                       240.0000000  254.76923  6.15%     -    0s
     0     0  254.76923    0    4  240.00000  254.76923  6.15%     -    0s

Explored 1 nodes (8 simplex iterations) in 0.04 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 240 153

Optimal solution found (tolerance 1.00e-04)
Best objective 2.400000000000e+02, best bound 2.400000000000e+02, gap 0.0000%
```

Table 10. Optimal solution to GAP.

```
Assign[Resource1,item3]: 1.0

Assign[Resource1,item4]: 1.0

Assign[Resource1,item7]: 1.0

Assign[Resource2,item5]: 1.0

Assign[Resource2,item6]: 1.0

Assign[Resource2,item8]: 1.0

Assign[Resource3,item1]: 1.0

Assign[Resource3,item2]: 1.0

Total profit: 240.0
```

Consider the execution of MTH with $f_{ij} = -b_{ij}$. The first phase of the algorithm gives:

$$b_{ij} = \begin{pmatrix} -22 & -14 & -10 & -6 & -8 & -16 & -6 & -25 \\ -21 & -9 & -19 & -26 & -7 & -7 & -10 & -7 \\ -17 & -17 & -19 & -25 & -12 & -12 & -17 & -19 \end{pmatrix}; \quad (K_j) = \begin{pmatrix} 27 \\ 26 \\ 35 \end{pmatrix}$$

As we can see item 4 has the maximum difference between the largest and the second largest f_{ij}, hence assign item 4 to the resource for which f_{ij} is a maximum, which is Resource 1. $\overline{K}$ denotes the remaining capacity after each assignment.

Iteration 1. i * = 4 : d* = 19, $y_4 = 1$, $\overline{K_1} = 21$; profit of assignment: 17

Iteration 2. i * = 8 : d* = 12, $y_8 = 2$, $\overline{K_2} = 19$; profit of assignment: 35

Iteration 3. i * = 3 : d* = 9, $y_3 = 1$, $\overline{K_1} = 11$; profit of assignment: 13

Iteration 4. i * = 1 : d* = +∞, $y_1 = 3$, $\overline{K_3} = 18$; profit of assignment: 35

Iteration 5. i * = 2 : d* = 8, $y_2 = 2$, $\overline{K_2} = 10$; profit of assignment: 6

Iteration 6. i * = 6 : d* = 5, $y_6 = 2$, $\overline{K_2} = 3$; profit of assignment: 26

Iteration 7. i * = 7 : d* = 11, $y_7 = 1$, $\overline{K_1} = 5$; profit of assignment: 42

Iteration 8. i * = 5 : d* = +∞ , $y_5 = 3$, $\overline{K_3} = 6$; profit of assignment: 20

Hence z^h =194, y_i = [3 2 1 1 3 2 1 2], ($\bar{K}$)= [5 3 6]

The second part of the algorithm applies the local exchange;

- y_i = [3 **2** 1 1 **3** 2 1 2],
- y_i = [3 **3** 1 1 **2** 2 1 2];

Hence, the solution is z^h =240, y_i = [3 3 1 1 2 2 1 2]. As we can see we obtain the optimal solution through this heuristic approach.

3.4 Multiple Knapsack Problem

The knapsack problem is encountered in the context of resource allocation with financial constraints. For example, how we determine what things to purchase given a fixed budget is an example of a knapsack problem. Everything has both cost and value; therefore, we look for the most value for the given cost. The classical knapsack problem is the problem of assigning a subset of *n* items to a fixed-size knapsack, such that the knapsack must be filled with the most useful and portable items. On the other hand, in the multiple knapsack problem, we consider assigning a subset of *n* items to *m* distinct knapsacks such that we wish to find the maximum profit of selected items by considering capacity of each knapsack. Thus, if we change the objective function to maximization in generalized assignment problem, and set the constraint (2) to less-than-or-equal to (≤), which means that each item can be in at most one knapsack; then, the resulting formulation generalizes the multiple knapsack problem. Consider a set of *n* items and a set of *m* knapsacks ($m \leq n$), p_i represents the profit of item *i*, and b_i is the weight of item *i* and K_j is the capacity of knapsack *j*. Therefore, multiple knapsack problem is formulated such that:

$$Maximize \qquad \sum_{i \in I} \sum_{j \in J} p_i x_{ij}$$

$$Subject\ to:$$

$$\sum_{i \in I} b_i x_{ij} \leq K_j \qquad j \in J, \qquad (1)$$

$$\sum_{j \in J} x_{ij} \leq 1, \qquad i \in I, \qquad (2)$$

$$x_{ij} \in \{0,1\}, \qquad i \in I,\ j \in J. \qquad (3)$$

where

$$x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is assigned to knapsack j;} \\ 0 & \text{o/w.} \end{cases}$$

Example 4.

Consider a multiple knapsack problem in which there are 9 items and 2 knapsacks, and profit and weight of each item is given as follows:

p_i = (160, 40, 120, 80, 120, 120, 130, 50, 60);

b_i = (40, 10, 40, 30, 50, 50, 55, 25, 40);

K_j = (100, 150)

We can solve this problem by using Gurobi Python API, and the output of the solver is shown below. The optimal solution of the Example 4. is 700. Note that If the problem size increases, the optimality gap is likely to increase as the problem is NP-Hard.

Table 11. Gurobi output for the multiple knapsack problem.

```
Total packed value: 700.0
Total packed weight: 250


Knapsak  1

Item 1 - weight: 40  value: 160
Item 2 - weight: 10  value: 40
Item 5 - weight: 50  value: 120
Packed knapsak weight: 100
Packed knapsack value: 320

Knapsak  2

Item 3 - weight: 40  value: 120
Item 4 - weight: 30  value: 80
Item 7 - weight: 55  value: 130
Item 8 - weight: 25  value: 50
Packed knapsak weight: 150
Packed knapsack value: 380
```

3.4.1 Martello And Toth Heuristic Method (MTHM) For Multiple Knapsack Problem

Martello and Toth (1990) proposed a heuristic technique in order to solve multiple knapsacks problem which is described as follows.

Step 1.

Create an items list, in such a way that (value[i] / weight[i]) is decreasing.

$\frac{p_1}{b_1} \geq \frac{p_2}{b_2} \geq \frac{p_3}{b_3} \ldots \geq \frac{p_n}{b_n}$, and

Sort the knapsacks so that

$K_1 \leq K_2 \ldots\ldots \leq K_m$

Step 2.

Insert the items, following the list order, to the first knapsack; then a set of remaining items will be inserted to the second knapsack, and this procedure is continued till the mth knapsack as long as the capacity constraint is violated.

Step 3.

The initial solution is improved through swapping each pair of items that are inserted to different knapsacks, and try to insert a new item in order to increase the total profit.

Step 4.

Try to exclude an inserted item in turn and replace it with a remaining item in order to increase the total profit.

Consider applying MTHM on Example 4. After applying Step 1, we would have the following;

$\frac{p_1}{b_1} \geq \frac{p_2}{b_2} \geq \frac{p_3}{b_3} \geq \frac{p_4}{b_4} \geq \frac{p_5}{b_5} \geq \frac{p_6}{b_6} \geq \frac{p_7}{b_7} \geq \frac{p_8}{b_8} \geq \frac{p_9}{b_9}$ and $K_1 < K_2$.

After applying the Step 2, we would obtain the following;

z = 640

(y_i) = (1, 1, 1, 2, 2, 2, 0, 0, 0) which represent;

$$y_i = \begin{cases} 0 \; if \; the \; item \; i \; is \; currently \; unassigned; \\ index \; of \; the \; knapsack \; it \; is \; assigned \; to, otherwise. \end{cases}$$

Step 3 produces the following solution firstly; y_i = (2, 1, 2, 1, 2, 1, 0, 0, 0) and then produces y_i = (1, 1, 2, 2, 2, 1, 0, 2, 0) respectively with a total profit of z= 690.

In Step 4, we can exclude item 5 and insert item 7 to the knapsack 2, and obtain the following optimal solution;

z=700

y_i = (1, 1, 2, 2, 0, 1, 2, 2, 0).

Thus, we obtained the optimal solution for Example 4 with MTHM. However, Lalami et al. (2012) argue that the main disadvantage of MTHM is that only the exchanges between a pair of items is regarded rather than combinations of items.

3.5 The Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) is also among the essential combinatorial optimization problems proposed by Koopmans and Beckmann in 1957. The problem is designed as a mathematical model for the location of a set of indivisible economical activities (Koopman & Beckman, 1957). We attempt to determine an allocation of a number of *n* facilities and to a number of *n* locations. A distance is specified for each pair of locations, and also a weight or flow is specified for each pair of facilities. The objective function is defined as assigning all facilities to different locations to find the minimum cost. Given *n* facilities and *n* locations, the cost of assigning facility *i* to location *j* and of facility *k* to location *l* is $f_{ik}d_{jl}$ with f_{ik} stands for the amount of flow between facilities *i* and *k* and d_{jl} indicates the distance between locations *j* and *l*. We can define a binary variable, say, x_{ij} =1 if facility *i* is assigned to location *j*, and 0 otherwise. Therefore, we can formulate the QAP such that:

$$Minimize \quad \sum_{i,j,k,l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}$$

$$Subject\ to \quad \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in \{1, \dots, n\} \qquad (1)$$

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

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in \{1, \dots, n\} \qquad (3)$$

Example 5.

Consider a facility location problem with five facilities (i=1,2,3,4,5) and five locations (j=A,B,C,D,E) whose flow and distance matrix tables given below. The purpose of the QAP is to assign all facilities to locations ensuring each facility is assigned to only one location while minimizing the total cost.

Table 1. The flow between facilities.

	1	2	3	4	5
1	0	0	2	0	3
2	0	0	0	3	0
3	2	0	0	0	0
4	0	3	0	0	1
5	3	0	0	1	0

Table 2. Distance between locations.

	A	B	C	D	E
A	0	50	50	94	50
B	50	0	22	50	36
C	50	22	0	44	14
D	94	50	44	0	50
E	50	36	14	50	0

We can write an example of valid assignment as the permutation p= [A, B, C, D, E] that is to say facility 1 is assigned to location A, facility 2 is assigned to location B, and facility 3 is assigned to location C and facility 4 is assigned to location D and facility 5 is assigned to location E. Our cost function is defined as = flow*distance matrix, we can calculate the objective function value as follows:

$$[\, f(1,2) * d(A,B) + f(1,3) * d(A,C) + f(1,4) * d(A,D) + f(1,5) * d(A,E) + f(2,3) * d(B,C) + f(2,4) * d(B,D) + f(2,5) * d(B,E) + f(3,4) * d(C,D) + f(3,5) * d(C,E) + f(4,5) * d(D,E) \,] * 2\ (symmetric\ matrix) = 900.$$ (which is not an optimal solution.)

3.5.1 A Heuristic for QAP: GRASP (Greedy Randomized Adaptive Search Procedure)

Greedy randomized adaptive search procedure (GRASP) has been established by Feo and Resende and applied to many difficult combinatorial optimization problems which can be defined as a combination of greedy elements with random search elements in a two-phase heuristic. More specifically, GRASP comprises of a construction phase and a local improvement phase. A complete candidate solution is constructed using a heuristic function in the construction phase, and then the neighborhood of the solution obtained in the first phase is tried to be improved in the local improvement phase until a stopping criterion is met. The pseudocode of GRASP is illustrated in the figure below.

Pseudo-code for GRASP.

```
procedure GRASP(RCLSize,MaxIter,RandomSeed)
1  InputInstance();
2  do k = 1,...,MaxIter →
3     ConstructGreedyRandomizedSolution(RCLSize,RandomSeed);
4     LocalSearch(BestSolutionFound);
5     UpdateSolution(BestSolutionFound);
6  od;
7  return BestSolutionFound
end GRASP;
```

Source: Burkard, R. E., Cela, E., Pardalos, P. M., & Pitsoulis, L. S. (1998). The quadratic assignment problem. In Handbook of combinatorial optimization (pp. 1713-1809). Springer, Boston, MA.

3.5.2 Obtaining solution for Example 5 with GRASP approach

In section, we described the GRASP method and how to implement it to the QAP.

3.5.3 Construction Phase

A candidate solution is obtained with a greedy heuristic approach with the following steps:

1. Arrange all facilities according to decreasing sum of material flow.
2. Arrange all locations according to increasing distance to all other locations.
3. Match the position of the largest value to the position of the smallest value.

Step 1.

Flow matrix

F	1	2	3	4	5	Σ
1	-	0	2	0	3	5
2	0	-	0	3	0	3
3	2	0	-	0	0	2
4	0	3	0	-	1	4
5	3	0	0	1	-	4

5
3
2
4
4
} **Manhattan-distance between facilities.**

Matrix is symmetric → consideration of the triangle matrix is sufficient.

The sequence of facilities:

1	4	5	2	3

Step 2.

Distance Matrix

L	A	B	C	D	E	Σ
A	-	50	50	94	50	244
B		-	22	50	36	158
C			-	44	14	130
D				-	50	238
E					-	150

The sequence of locations:

C	E	B	D	A

Step 3.

1	4	5	2	3
C	E	B	D	A

Assignment: p= [C, D, A, E, B] which is the greedy heuristic solution.

1	**2**	**3**	**4**	**5**
C	**D**	**A**	**E**	**B**

Total Cost

	1	2	3	4	5	Σ	*Total Cost*
1	-	(0*44)	(2*50) *	(0*14)	(3*22)		352*2 (symmetric)
2		-	(0*94)	(3*50)	(0*50)		
3			-	(0*50)	(0*50)	**352**	= **704**
4				-	(1*36)		
5					-		

* Facility 1 and 3 assigned to C and A. Distance between C and A: 50, and Flow between 1 and 3 is: 2.

3.5.4 Local Search Phase

A pairwise exchange approach is applied which works as follows: Select randomly two locations and swap their position in the given solution, and if this modification results in a better cost, then we need to update the current solution. The algorithm continues till no more improvements can be found.

Greedy heuristic solution: [C, D, A, E, B] now apply pairwise exchange algorithm:

Iteration 1. [**D, C**, A, E, B]:

	1	2	3	4	5	Σ	*Total Cost*
1	-	0*44	2*94	0*50	3*50		416*2 (symmetric)
2		-	0*50	3*14	0*22		
3			-	0*50	0*50	**416**	= **832**
4				-	1*36		
5					-		

Do not accept the solution because the total cost is greater than the greedy heuristic solution.

Iteration 2. **[B**, D, A, E, **C**]:

	1	2	3	4	5	***Σ***	***Total Cost***
1	-	0*50	2*50	0*36	3*22		330*2 (symmetric)
2		-	0*94	3*50	0*44		
3			-	0*50	0*50	**330**	= **660**
4				-	1*14		
5					-		

Accept the solution and update it as the current solution. Continue until no more improvements are found.

The solution obtained with local search is **p = [E, D, A, B, C]** with a total cost of 628.

The optimal solution obtained by a QAP solver is 628 with **p = [E, D, A, B, C].**

	1	2	3	4	5	***Σ***	***Total Cost***
1	-	0*50	2*50	0*36	3*14		314*2 (symmetric)
2		-	0*94	3*50	0*44		
3			-	0*50	0*50	**314**	= **628**
4				-	1*22		
5					-		

4. SOLUTION METHOD

It is mostly intractable to solve difficult combinatorial optimization problems optimally, therefore one needs to be satisfied with the near-optimal solutions that are obtained by heuristic algorithms. Heuristic algorithms are widely categorized as constructive and improvement algorithms. There is another class of heuristic algorithms, named metaheuristics, that could be utilized to solve a large class of combinatorial optimization problems either directly or with minor modifications (Duman et al., 2012). In this study, we designed a Migrating Birds Optimization (MBO) which is a nature-inspired metaheuristic algorithm proposed by Duman et al. (2012), and a hybrid Simulated Annealing and Tabu Search algorithm to solve the GAP of BinYaprak online education website where an expert can be assigned to more than one learner ensuring each learner is assigned at most one expert subject to each expert's capacity.

4.1 Migrating Birds Optimization Algorithm (MBO)

The MBO is first designed to solve the quadratic assignment problem (QAP) arising from printed circuit board assembly workshops by Duman et al. (2012), and the numerical analyses show that the MBO algorithm outperforms the Simulated Annealing approximately three percent on average. Duman and Eliküçük (2013) solved the credit card fraud detection problem by using the MBO algorithm with some modifications inspired by the genetic algorithms, and they found out that modified MBO outperforms the standard MBO. Öz and Öz (2020) redesigned the implementation of the MBO algorithm, and proposed a scalable parallel MBO, PMBO, to solve multi-objective task

allocation problem. Ülker and Tongur (2016) present an implementation of MBO to solve 1-0 knapsack problems, and claim that the MBO algorithm is fairly reasonable for 0-1 knapsack problems in terms of running time. Tongur et al. (2020) used MBO to solve the land distribution problem, and found out that the land distribution is done faster than the conventional method, thus reducing the project cost. Ruiz et al. (2017) proposed Multi-Leader MBO to solve printed circuit board problems, and computational results indicate that the algorithm is quite competitive, and provides new best solutions. Soto et al. (2018) address the problem of Machine-Part Cell Formation by implementing MBO, and allege that MBO provides better results for the problem when it is compared to other methods reported in the literature. Moreover, Sioud and Cagné (2018) introduced Enhanced Migrating Birds Optimization Algorithm (EMBO) so as to solve the permutation flow shop problem, and the computational experiments demonstrate that EMBO provides state-of-the-art outcomes when compared to results obtained by other algorithms. Lastly, Aksakallı et al. (2018) proposed hybridization of Migrating Birds Optimization with Simulated Annealing which is specifically designed so that the algorithm can escape the local optima. The authors tested the hybrid MBO on Quadratic Assignment Problem instances, and noted that hybrid MBO outperforms the standard MBO by approximately two-thirds of the QAP test instances.

The Migrating Birds Optimization (MBO) algorithm is motivated by the V formation flight of the migratory birds which is a very effective formation in energy minimization during migration.

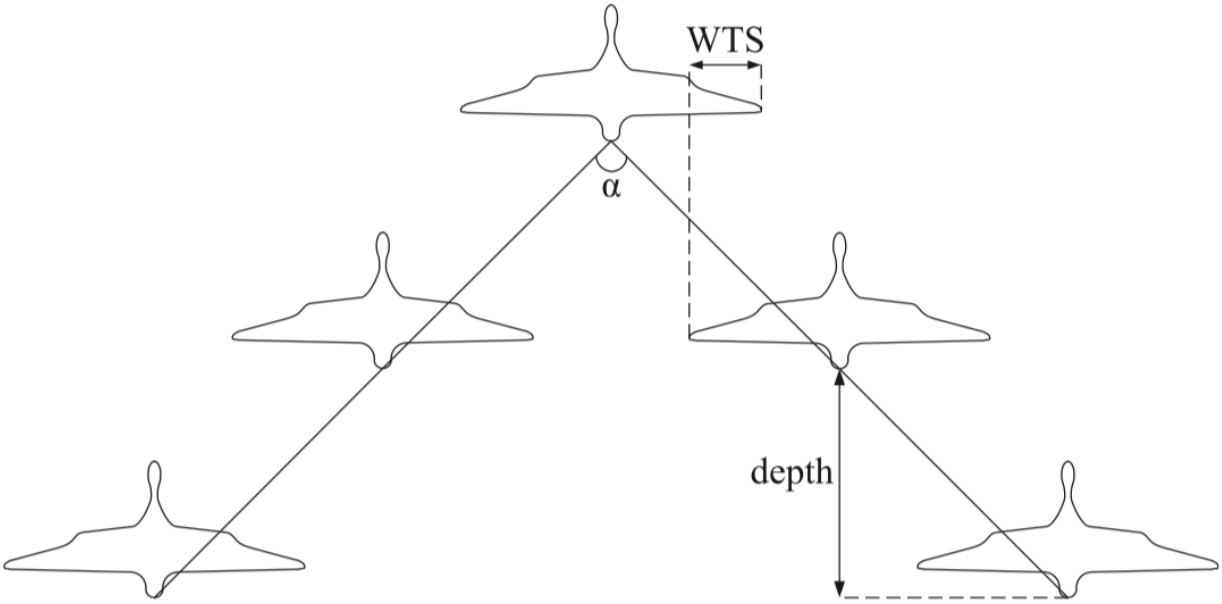

Figure 4: The V formation.

Source: Duman et al. (2012).

The leader bird is assumed to spend the most energy during migration as shown in the V formation figure above. The birds in the other positions get benefit from the birds in front. While it seems logical that energy savings increases as we move towards back in the right or left side of the flock; yet, to the best of our knowledge, there is not a study in the literature to support this hypothesis. Duman et al. (2012) argue that the energy saving of the birds that follow the leader bird is either same or relatively higher for the birds in the middle part of the flock compared to the leader bird. Moreover, it is supposed that the leader bird goes to the end of the flocks when it gets tired after flying a while, and one of the birds becomes the leader, and thus, the birds keep the V formation of the flock during migration.

The MBO starts with *n* initial solutions which are carried by a flock of n birds in a V formation just as migratory birds' flocks. The algorithm is initialized with the first solution, which is the leader bird, after that those initial solutions are tried to be improved on each iteration through its neighbor solutions by progressing on the lines towards the end of each tail. If the best neighbor solution provides an improvement, then the current solution is replaced by that best neighbor solution. The MBO neighborhood structure has a benefit mechanism in which the best-unused neighbor solutions are shared with other solutions (birds). After all solutions are either improved or tried to be improved by their neighbor solutions, this procedure is repeated a couple of times, which is the tour number (m). Then, imitating the natural behavior of migrating flock, the leader bird goes at the back of the either left or right tail of the flock, which allows us to explore the different parts of search space, and then another loop begins. After a number of iterations (K), the MBO algorithm is stopped ensuring at least one leadership for all the birds in the flock, and the best solution in the flock is selected.

The parameters of the MBO algorithm are as follows:

Table 12. List of Notation

Parameters	
n	The number of initial solutions
k	The number of neighbor solutions
x	The number of neighbor solutions to be shared
m	Tour number
K	Iteration limit

4.2 Initial Population and Solution Representation

Firstly, *n* initial solutions are generated randomly and placed on a V formation randomly as well. Initial solutions are generated as follows: a learner is selected randomly, and then the randomly selected learner is assigned to each expert in order considering the expert's capacity. A neighbor solution is obtained by swapping any random two learners assigned to different experts.

Experts	1	1	2	3	4	4	4	5	5	5
Learners	3	4	5	6	8	9	10	2	7	1

Figure 5: Configuration of a feasible solution of MBO.

Experts	1	1	2	3	4	4	4	5	5	5
Learners	3	4	**10**	6	8	9	**5**	2	7	1

Figure 6: Neighbor generation with swap operator.

An illustration of the V formation of the initial population (n=7) in the MBO is presented as follows:

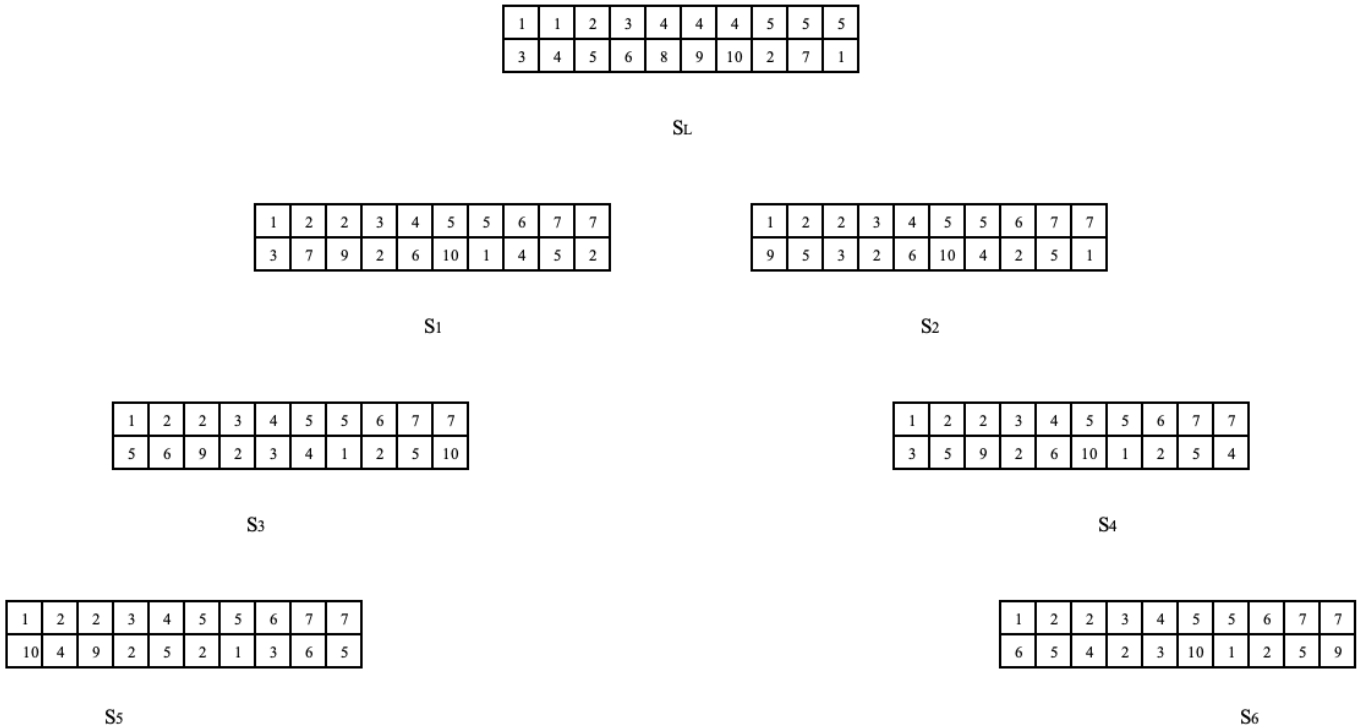

Figure 7: V formation of the initial solution.

4.3 Numerical Experiment

MBO has been tested on a computer with 64-bit Windows 2.00GHz processor Intel Core i7 and with 16 GB RAM, and coded with PyCharm Professional 2020.3 language, and also, GUROBI Optimizer 9.1 is used in Python API Interface to solve the mathematical model for GAP of BinYaprak Online Education Website. Consider the following real example of 15 experts and 23 learners of the BinYaprak online website with the following parameters: n = 5, k = 3, x = 1, m=2, K=10. The following table provides solutions for the first two tours and the first iteration of the algorithm to give an idea of how the algorithm works. After two tours, the leader change procedure is applied. The leader bird goes at the end of the randomly selected right or left tail, and one of the randomly chosen birds (S_1 or S_2) becomes the new leader. And, the algorithm is stopped after 10 iterations (K).

Table 13. Neighbor solutions of each bird for n=5, k=3 and x=1.

Tour 1					**Tour 2**
Bird	**Fitness Value**	**Neighbor**	**Fitness Value**	**Used Neighbor**	**Neighbor**
$\mathbf{S_L}$	615	N_{01} N_{02} N_{03}	642 591 588	**642**	**666** 597 594
$\mathbf{S_1}$	603	N_{11} (Shared=N_{02}) N_{12} N_{13}	591 591 549	**603**	**603** 597 552
$\mathbf{S_2}$	627	N_{21} N_{22} N_{23} (Shared=N_{03})	675 651 588	**675**	654 648 594
$\mathbf{S_3}$	621	N_{31} N_{32} N_{33} (Shared=N_{12})	627 594 591	**627**	**654** 603 600
$\mathbf{S_4}$	774	N_{41} N_{42} N_{43} (Shared=N_{22})	693 666 651	**774**	768 720 654

After two tours (m), the leader change procedure is applied and this is repeated ten times (K). Table 14 shows the solution found by MBO for the instance in which n = 5, k = 3, x = 1, m=2, K=10.

Table 14. Solution for the numerical example.

Bird	**Fitness Value**	**Assignment Solution**
S_L	**1230***	[['L20', 'L1', 'L12', 'L22'], ['L8'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13', 'L9'], ['L5', 'L10'], ['L23', 'L18'], ['L15', 'L11'], ['L7', 'L16', 'L4'], ['L6', 'L21'], ['L19'], []]
S_1	1209	[['L20', 'L1', 'L12', 'L22'], ['L8'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13', 'L9', 'L18'], ['L5', 'L10'], [], ['L15', 'L11'], ['L7', 'L4', 'L16'], ['L21', 'L6'], ['L19'], ['L23']]
S_2	**1230***	[['L20', 'L1', 'L12', 'L22'], ['L8'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13', 'L9'], ['L5', 'L10'], ['L23', 'L18'], ['L15', 'L11'], ['L7', 'L16', 'L4'], ['L6', '21'], ['L19'], []]
S_3	**1230***	[['L20', 'L1', 'L12', 'L22'], ['L8'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13', 'L9'], ['L5', 'L10'], ['L23', 'L18'], ['L15', 'L11'], ['L7', 'L16', 'L4'], ['L6', 'L21'], ['L19'], []]
S_4	1215	[['L20', 'L1', 'L12', 'L22'], ['L8', 'L13'], ['L3', 'L9'], ['L14'], ['L17'], ['L2'], [], [], ['L5', 'L10'], ['L23', 'L18'], ['L15', 'L11'], ['L7', 'L16', 'L4'], ['L6', 'L21'], ['L19'], []]

We can see from the table that birds S_L, S_2, S_3 achieved an optimal solution for the problem with a 06.94 (s) CPU time.

4.4 The Simulated Annealing (SA) and Tabu Search (TS) Algorithm

Simulated Annealing (SA) is one of the well-known metaheuristic algorithms which is based on the analogy between the physical annealing of solid materials and the problem of solving both combinatorial and global optimization problems with a strategy to avoid local optima. The interest in SA started with Metropolis et. al (1953) who introduced SA to find out the equilibrium configuration of a collection of atoms at a given temperature, after that SA was developed as an optimization technique by Kirkpatrick et al. (1983). The quintessential feature of SA is allowing moves that result in a solution of worse quality than the current solution so as to escape from the local minima. Basically, the SA iteratively selects a candidate solution randomly in the local neighborhood, and if it improves the current solution, the candidate solution is replaced with the current solution. Otherwise, the probability of accepting a non-improving solution is controlled by a temperature parameter T, and it is updated by a deterministic cooling parameter α. That is to say, the probability of accepting degraded solutions is calculated by applying a Metropolis criterion which is $(\delta) = e^{\frac{-(f(x)-f(0))}{T}}$. More specifically, the difference between the candidate solution and the current solution is divided by the current temperature. We can apply the Metropolis criterion by generating a random x uniformly in the range (0,1), and if $x < e^{\frac{-(f(x)-f(0))}{T}}$, then we can accept the non-improving solution. While designing a Simulated Annealing Algorithm, the initial temperature needs to be set high enough as the system should be melted completely, then should progressively be decreased since the search proceeds till the system reaches its freezing point.

Tabu Search Algorithm (TS) is one of the best-known metaheuristics proposed by Glover (1977) for integer programming by surrogate constraints. It is important to note that the term "metaheuristic" was used for the first time in the literature with Tabu Search Algorithm (Resende et al, 2017). Tabu Search Algorithm is one of the S-metaheuristics (single-solution based metaheuristic) which starts with an initial solution and searches through the solution space by making transformations to shift from one solution to the next. Other popular S-metaheuristics are Simulated Annealing and Variable Neighborhood Search etc. On the other hand, Genetic Algorithm (GA), Migrating Birds Algorithm (MBO) and Particle Swarm Optimization are all P-Metaheuristics (population-based metaheuristics) which start with multiple initial solutions and try to improve them. As a matter of fact, TS behaves just like a steepest Local Search algorithm; however, TS accepts non-improving solutions in order to avoid local optima if all the neighbor solutions are non-improving. If a better neighbor solution is found, then it is replaced by the current solution. The typical characteristic of TS is that it uses a memory structure to store information about the search history so that cycles are avoided. More specifically, TS keeps a memory of moves applied recently which is termed as tabu list, and it composes the short-term memory in order to avoid the search from revisiting already visited solutions before. Moreover, to determine when to override the tabu activation rule, an aspiration criterion is introduced in TS. The most popular aspiration criterion is allowing a tabu move when a tabu move would result in a solution better than any visited so far.

There is a recent trend in TS which is the use of medium-term memory and long-term memory in order to intensify and diversify the search into regions that are not explored yet. In the intensification mechanism, the medium-term memory keeps the elite (e.g., the most promising) solutions that have been found during the search, and prioritizes the

attributes of the set of good solutions employing weighted probability. On the other hand, in the diversification mechanism, the long-term memory is used to store information about the visited solutions during the search. The idea is to force the search into the non-explored region of the search space. As explained in (Gendreau & Potvin, 2010), there are three popular diversification strategies which are as follows:

1. **Restart Diversification:** This approach is based on introducing in the current or best solution the least visited components, after that, a new search starts from this point.

2. **Continuous diversification:** In this strategy, the historical information is also used, yet in a continuous manner. This approach progressively incorporates diversifying effect into the usual searching process by biasing the evaluation of neighbor solutions. This is achieved in such a way that a small term, which is related to solution component frequencies, is added to the objective.

3. **Strategic oscillation:** This strategy considers and penalizes the intermediate solutions which are infeasible. The strategic oscillation guides the search against infeasible solutions, then gets back to a feasible solution.

4.5 Hybrid SA/TS Algorithm Description

In this section, we describe the details of the hybrid SA/TS algorithm that we designed to solve the GAP of BinYaprak online education website. There are several hybrid SA/TS approaches in the literature. For instance, Osman (1995) designed a hybrid simulated annealing and tabu search algorithm to solve the Generalized Assignment Problem. He pointed out that hybrid SA/TS outperforms set partitioning, simulated annealing and curtailed branch and bound with regard to solution quality and computational effort. Furthermore, Küçükoğlu et al. (2019) proposed a hybrid simulated annealing and tabu search method to solve electric traveling salesman problem with time windows and mixed charging rates, and authors reported that significant distance savings achieved through charging the electric vehicle at customer locations. Kaviani et al. (2014) designed a hybrid simulated annealing and tabu search method for Quadratic Assignment Problem (QAP), and the results of the proposed method indicate that the hybrid approach is able to solve real-world problems efficiently when compared to Tabu Search.

The hybrid SA/TS algorithm that we designed takes the advantage of Simulated Annealing and Tabu Search to escape from the local maxima by integrating the solution acceptance procedure of Simulated Annealing and tabu list structure of the Tabu Search algorithm. In order to enhance the performance of hybrid SA/TS, we introduced a long-term memory to encourage the diversification of the search in which the entire previous search history is used to guide the search to find the next unvisited region of the search space. We designed and implemented a *restart diversification* strategy. This strategy keeps the information of each applied move and each visited solution during the search to avoid cycling, and if an improving solution cannot be found after a number of iterations, for instance, in some situations the memory guided search seems to be trapped

in a local optimum region, the memory is reset and search is restarted from the current solution, and local search procedure is applied. The aim is to force the search into the non-explored region of the search space. In the hybrid SA/TS, the initial solution is generated randomly such that an expert and a learner is selected randomly, then the learner is assigned to that randomly selected expert. A neighbor solution is obtained by swapping randomly selected two learners assigned to different experts. Then, the hybrid Simulated Annealing and Tabu Search Algorithm with the following parameters performed. The initial temperature is set to $T_0 = 1000$, and the final temperature to $T_F = 0.01$, and a geometric cooling schedule with a cooling rate of 0.99 with the standard epoch length: 2. The solution is accepted if it is improved, otherwise, the probability of accepting a new solution is calculated with the Metropolis criterion, which accepts non-improving solutions with a probability which is computed by $(\delta) = e^{\frac{-(f(x)-f(0))}{T}}$. Furthermore, if an improving solution cannot be found in 50 consecutive iterations, a new search is restarted from the current solution, and the algorithm is stopped when the temperature achieves $T_F = 0.01$. The hybrid SA/TS is run 5 times.

4.6 Numerical Experiment

The hybrid SA/TS has been tested on a computer with 64-bit Windows Server with 2.00GHz processor Intel Core i7 and with 16 GB RAM, and the hybrid SA/TS coded with PyCharm Professional 2020.3 language, and also GUROBI Optimizer 9.1 is used in Python API Interface to solve the mathematical model for GAP of BinYaprak Online Education Website. Consider the same instance solved by MBO. Recall that there are 15 experts and 23 learners of the BinYaprak online website, and the aim is to assign each

learner to an expert based on the capacity of each expert. The objective function maximizes the total assignment score.

Table 15. Solution of the Hybrid SA/TS.

Run	Objective Function Value	Assignment Solution	CPU Time (s)
1st Run	1065	[['L12', 'L22', 'L16', 'L20'], ['L1'], ['L3'], ['L14'], ['L17'], ['L8'], [], ['L13', 'L19', 'L18'], ['L10', 'L5', 'L11'], ['L2'], ['L15'], ['L4', 'L7'], ['L21', 'L6'], ['L9'], ['L23']]	3.28
2nd Run	1212	[['L12', 'L1', 'L20', 'L8'], [], ['L3'], ['L14'], ['L17'], ['L2'], ['L22'], ['L13'], ['L5', 'L10'], ['L18', 'L23'], ['L15', 'L11'], ['L4', 'L7', 'L16'], ['L21', 'L6'], ['L19', 'L9'], []]	5.58
3rd Run	**1230***	[['L1', 'L12', 'L20', 'L22'], ['L8', 'L7'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13'], ['L5', 'L10'], ['L18', 'L23'], ['L15', 'L11'], ['L16', 'L4'], ['L21', 'L6'], ['L19', 'L9'], []]	3.59
4th Run	**1230***	[['L1', 'L12', 'L20', 'L22'], ['L8', 'L7'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13'], ['L5', 'L10'], ['L18', 'L23'], ['L15', 'L11'], ['L16', 'L4'], ['L21', 'L6'], ['L19', 'L9'], []]	4.14
5th Run	**1230***	[['L1', 'L12', 'L20', 'L22'], ['L8', 'L7'], ['L3'], ['L14'], ['L17'], ['L2'], [], ['L13'], ['L5', 'L10'], ['L18', 'L23'], ['L15', 'L11'], ['L16', 'L4'], ['L21', 'L6'], ['L19', 'L9'], []]	7.26

* Optimal solution for the problem.

4.7 Hybrid SA/TS and Variants

In this part, we present the implementation of hybrid SA/TS with different parameters on small and large instances to obtain the best parameters of the hybrid SA/TS which might influence the performance of the algorithm. In our computational study, we examine two variants of the hybrid SA/TS. We tested the Hybrid SA/TS_1 with the following parameters; T_0=1000, epoch length is five and the geometric schedule with a cooling rate of 0.90. We tested the Hybrid SA/TS_2 with the following parameters; T_0 = 10000, epoch length is 2 and geometric schedule with a cooling rate of 0.99. Each instance is run 10 times, and we put a time limit of 60 s for the instances 1,2,3,4, 5, 6 and 150 s for the instances 7 and 8 and 1 hour for the instances 9 and 10. The results of both variants are provided in Table 16 below. Finally, restart diversification is implemented such that if an improving solution cannot be found after 50 consecutive iterations, a new search is restarted from the current solution for the hybrid SA/TS_1, and SA/TS_2.

Table 16. Comparison of Hybrid SA/TS_1, Hybrid SA/TS_2 and Gurobi Solutions.

			Hybrid SA/TS_1						**Hybrid SA/TS_2**					
			Assignment Score			**Gap (%)**			**Assignment Score**			**Gap (%)**		
Inst.	**j**	**i**	**min**	**max**	**avg**	**min**	**max**	**avg**	**min**	**max**	**avg**	**min**	**max**	**avg**
1	5	10	**210***	**210***	**210***	0.00	0.00	0.00	**210***	**210***	**210***	0.00	0.00	0.00
2	5	20	**549***	**549***	**549***	0.00	0.00	0.00	**549***	**549***	**549***	0.00	0.00	0.00
3	10	30	**987***	**987***	**987***	0.00	0.00	0.00	**987***	**987***	**987***	0.00	0.00	0.00
4	15	30	**1230**	**1230***	**1230***	0.00	0.00	0.00	**1230**	**1230***	**1230***	0.00	0.00	0.00
5	15	50	2466	2507	2452	0.98	3.59	2.39	**2532***	**2532***	**2532***	0.00	0.00	0.00
6	20	100	4149	4305	4226	3.17	6.68	4.93	4350	4428	4401	0.40	2.14	1.01
7	20	200	4874	5008	4974	6.11	8.62	6.75	5070	5139	5127	2.92	4.94	3.88
8	20	250	6051	6187	6212	7.56	9.59	7.18	6255	6474	6321	3.27	6.54	5.55
9	30	300	7103	7367	7320	2.89	3.76	3.51	7500	**7587***	7467	0.00	1.15	0.89
10	50	500	13449	13662	13598	3.72	5.22	4.17	13992	**14190***	14124	0.00	1.40	0.93

*: Optimal Solution

Table 16 compares the solution values obtained by hybrid SA/TS_1 and hybrid SA/TS_2 to decide on the best parameters of hybrid SA/TS. We also report the optimality gap by using Gurobi solutions. We can clearly see that both hybrid SA/TS variants can quickly provide high quality solutions. However, for the larger instances, hybrid SA/TS_2 performs better than hybrid SA/TS_1. Therefore, the hybrid SA/TS_2 with the following parameters

T_0 = 10000, epoch length is two and geometric schedule with a cooling rate of 0.99 will be used to compare performances of hybrid SA/TS_2 and MBO.

4.8 Parameter Fine Tuning for MBO

In this section, the following parameters are tested on instances 4 and 10 which include 15 experts and 30 learners, and 50 experts and 500 learners respectively to find the best parameters of MBO. Each instance is run 10 times, and the Gurobi solver was also used to solve mathematical model of GAP, and the optimal solutions **1230*** and **14190*** were obtained for instances 4 and 10 respectively. Table 17 provides the objective function value of GAP with the following parameters. The parameter x is set to be 1 due to the fact that in the case of x being 1, the best-unused neighbor of a solution is propagated to its immediate predecessor and it is used by that one or not but, it is depleted at that level; however, when x is greater than 1, some of them can also be propagated to the solutions further back which may in return result in a quicker convergence and worse performance (Duman et al., 2012).

Table 17. Initial analyses to find the best parameter values of MBO.

				(Instance 4)						(Instance 10)					
				Gap (%)			CPU Time (s)			Gap (%)			CPU Time (s)		
n	**k**	**x**	**m**	**min**	**max**	**avg**	**min**	**max**	**avg**	**min**	**max**	**avg**	**min**	**max**	**avg**
3	3	1	1	25.12	37.31	32.76	00.39	00.45	00.40	40.16	44.52	41.18	02.98	03.60	03.64
5	3	1	2	25.60	36.09	31.30	00.38	00.60	00.47	38.90	42.51	40.50	02.98	06.50	05.73
7	3	1	3	20.73	30.00	26.17	00.63	00.76	00.70	38.75	40.20	39.49	05.28	07.53	06.92
9	3	1	5	14.39	24.63	19.59	01.37	02.35	01.47	36.46	38.59	3763	22.02	24.06	23.60
13	3	1	5	4.55	12.68	8.37	03.52	04.17	03.68	35.09	37.06	36.24	110	120	107
19	3	1	6	0.73	7.31	3.65	08.26	10.13	06.30	32.93	38.90	32.93	96	136	110
25	3	1	7	0.00	0.00	0.00	25.37	31.24	27.52	30.76	33.12	31.37	435	550	495
51	**3**	**1**	**10**	0.00	0.00	0.00	107	124	111	3.28	8.56	4.63	1680	2880	2100

According to solutions obtained by testing different combinations of parameters, we can suggest that the parameters n=51, k=3, x=1, and m=10 can provide high-quality solutions, thus we will be using these parameters to compare the performance of MBO and the hybrid SA/TS_2 on small and large-sized instances in the next section.

4.9 Performance of the MBO, hybrid SA/TS and MTH on small and large-sized instances

To analyze the performance of MBO, hybrid SA/TS_2 and MTH, each instance is run 10 times, and we put a time limit of 60 s for the instances 1, 2, 3, 4, 5, 6 and 150 s for the instances 7 and 8 and 1 hour for the instances 9, 10 and 11 as explained in Wu et al. (2004) and Duman et. al (2012). The number of experts is denoted by j and the number of learners is denoted by i. Also, Gurobi Optimizer 9.1 is used in Python API Interface to solve the mathematical model to use it as a baseline reference against which we can compare our heuristic methods. The MBO and hybrid SA/TS_2 have been performed on the instances with the following parameters respectively; n=51, x=1, k=3, m=10 for MBO, and T_0 = 10000, and the cooling rate of 0.99, epoch length is 2 for, and if an improving solution cannot be found in 50 consecutive iterations, the search is restarted from the current solution.

Table 18. Comparison of MBO, hybrid SA/TS$_2$ and MTH performances on small and large sized instances.

			Gurobi		Hybrid SA/TS$_2$						MBO						MTH	
					Assignment Score			Gap (%)			Assignment Score			Gap (%)			Score	Gap (%)
#	j	i	Z_G	O.Gap	min	max	avg	min	max	avg	min	max	avg	min	max	avg	Z_{MTH}	Gap
1	5	10	210	0.00	**210***	**210***	**210***	0.00	0.00	0.00	**210***	**210***	**210***	0.00	0.00	0.00	**210***	0.00
2	5	20	549	0.00	**549***	**549***	**549***	0.00	0.00	0.00	**549***	**549***	**549***	0.00	0.00	0.00	**549***	0.00
3	10	30	987	0.00	**987***	**987***	**987***	0.00	0.00	0.00	**987***	**987***	**987***	0.00	0.00	0.00	**987***	0.00
4	15	30	1230	0.00	**1230***	**1230***	**1230***	0.00	0.00	0.00	**1230***	**1230***	**1230***	0.00	0.00	0.00	1224	0.48
5	15	50	2532	0.00	**2532***	**2532***	**2532***	0.00	0.00	0.00	2514	**2532***	2523	0.00	0.71	0.35	2523	1.10
6	20	100	4446	0.00	4350	4428	4401	0.40	2.14	1.01	4062	4104	4080	7.69	8.63	8.28	2504	2.36
7	20	200	5334	0.00	5070	5139	5127	2.92	4.94	3.88	4782	4833	4809	9.39	10.35	9.84	5065	5.04
8	20	250	6693	0.00	6255	6474	6321	3.27	6.54	5.55	5628	5886	5754	12.05	15.91	14.02	5601	16.31
9	30	300	7587	0.00	7500	**7587***	7467	0.00	1.15	0.89	7188	7275	7227	4.11	5.25	4.74	7219	4.85
10	50	500	14190	0.00	13992	**14190***	14124	0.00	1.40	0.93	14024	14058	14036	1.16	3.96	2.72	13786	2.84
11	1000	10000	349490	16.80	360150	380100	370117	9.53	14.90	11.91	349465	370150	358172	11.90	16.82	14.75	320156	23.80

*: Optimal Solution

Table 18 compares solutions obtained by MBO, hybrid SA/TS_2, MTH and Gurobi out of 10 runs for the 10 instances. We can clearly see that hybrid SA/TS_2 and MBO outperforms MTH algorithm in terms of the solution quality. Also, the hybrid SA/TS_2 performs better than MBO. Therefore, we can suggest that for the small instances 1, 2, 3, 4 and 5 hybrid SA/TS_2 and MBO can be used to obtain optimal solutions. On the other hand, when the data size increases, the optimality gaps of MBO for the five instances (instances 6, 7, 8, 9 and 10) are higher than that of hybrid SA/TS_2. Therefore, the hybrid SA/TS_2 can be used in practice to obtain high-quality solutions for larger sized data sets. Please note that, in order to obtain better results, one needs to run the MBO and hybrid SA/TS_2 for a longer time. What is more, for a very big data as the instance 11, hybrid SA/TS_2 and MBO performs better compared to Gurobi for the instance 11 in one-hour computational time. In order to calculate the optimality gap of Gurobi solver, hybrid SA/TS_2, MBO and MTH for the instance 11 in 1-hour computational time, the Gurobi solver is run around four hours (14.420 seconds) and obtained optimal solution of 420155. To sum up, we can use hybrid SA/TS_2 to obtain high quality solutions in the case of a larger instance of BinYaprak online education website.

5. DISCUSSION AND CONCLUSION

5.1 Discussion

The findings of this study suggests that our hybrid algorithm provides good solutions for BinYaprak online education website. We can clearly see from the Table 18 that Gurobi solver provides optimal solutions for the first ten instances. Thus, the following question may arise: “Why should we use heuristic to solve the problem of BinYaprak online education website?”. As a matter of fact, the website cannot afford to purchase the commercial license of Gurobi which costs $10000, this is a big financial burden for the website. There are some free optimization software available on the internet such as Google OR, however integrating those software into the website is not possible right now due to the software architecture of the website, therefore we need to develop our own algorithm which provides assignment of a learner to an expert each week. Since the BinYaprak does not want to purchase and use a commercial solver, we can provide with the website our hybrid algorithm or MBO to obtain assignments of learners. Furthermore, there are 10000 members registered on the website currently, so if we need to assign all members at one time, Gurobi solver does not provide optimal solution in one-hour computational time, and our hybrid algorithm performs better compared to Gurobi when we need to assign 10000 members at once, so in such a case, we can use our hybrid algorithm to solve BinYaprak problem.

This study has potential limitations that should be noted. First, since we maximize the total assignment score, our model assigns a learner, who have the highest score, to an expert. Each week highest scored learners are assigned based on the capacity of each

expert, so less scored learners are ignored, and our model does not give priority to those who have never been assigned before. Since the main purpose of the website is to create ways for content and network aggregation between experts and learners for the ease of knowledge discovery, if a learner has an assignment score, then he or she should be given an opportunity to benefit from a mentorship of an expert. Thus, improving our mathematical model to give priority those who have never been assigned but had an assignment score is the future work of this study.

Furthermore, our model assigns a learner to only one expert, although he or she might have different assignment scores with different experts. So, in the case of a learner who is not satisfied with the information he or she received from an expert or the mentorship of the expert, our model eliminates potential network aggregation between a learner and an expert. Thus, in order to tackle this problem, we can introduce a demand parameter d which represent the number of experts that a learner can be assigned based on assignment score. Therefore, we can assign a learner more than once. For example, if a learner is not satisfied with the mentorship of an expert, he or she can be satisfied with different mentorship by assigning he or she to different experts. So, we can formulate the model such that:

$$Maximize \qquad \sum_{l \in L} \sum_{e \in E} s_{el} x_{el}$$

$$Subject\ to:$$

$$\sum_{l \in L} a_{el} x_{el} \leq C_e \qquad e \in E, \tag{1}$$

$$\sum_{e \in E} x_{el} \leq d_l, \qquad l \in L, \tag{2}$$

$$x_{e,l} \in \{0,1\}, \qquad e \in E,\ l \in L. \tag{3}$$

We can obtain value of d_l from our score matrix. For example; L_1 has the 9 different assignment scores, hence d_1 is 9 for L_1. By assigning a learner more than once might provide learners to expand their networks; however, we would eliminate other learners who are less scored, and they cannot get benefit from the website. This problem was also discussed among the managers of the website, but the CEO of the website preferred assigning the always the one who has the highest score, since having the highest score may indicate that a learner has same or similar interests with an expert, thus the mentorship could be more fruitful.

E/L	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	C
E1	66	0	0	0	0	0	0	27	0	0	4
E2	18	0	0	0	0	0	90	27	0	0	3
E3	0	0	51	0	0	0	0	24	27	0	2
E4	27	0	0	0	0	0	0	0	0	0	3
E5	9	12	0	0	0	0	27	27	0	0	1
E6	15	96	27	27	27	0	0	27	0	0	1
E7	24	0	0	0	0	0	0	0	0	0	2
E8	15	18	0	0	0	0	0	0	27	0	3
E9	18	21	0	0	66	21	0	0	0	45	3
E10	60	45	0	21	21	0	0	0	0	0	2

5.2 Conclusion

In this study, we analyzed the assignment problem of the BinYaprak online education website in which we wish to achieve the maximum score assignment of a number of learners to a number of experts which satisfies both sides as much as possible. In our context, an expert might be assigned to multiple learners ensuring each learner is assigned at most one expert not exceeding the capacity of the expert is a feasible solution, hence, the problem becomes a generalized assignment problem (GAP) which is known to be NP-Hard. In order to solve the GAP of the BinYaprak website, we designed a new nature-inspired metaheuristic called Migrating Birds Optimization algorithm and a hybrid Simulated Annealing and Tabu Search algorithm with a restart diversification strategy, also used a well-known algorithm designed by Martello and Toth (MTH) to solve generalized assignment problem. We tested algorithms on small and large-sized instances. Also, Gurobi Optimizer 9.1 was used in Python API Interface to use it as a baseline reference against which we can compare our heuristic methods. Our numerical analyses indicate that the both proposed hybrid SA/TS_2 and MBO algorithm provide quick and high-quality solutions for small-sized instances and outperforms MTH; however, when data size increases, the hybrid SA/TS_2 outperforms the MBO with regards to the solution quality. Hybridization of MBO with Tabu Search algorithm by integrating the tabu list structure of TS with intensification and diversification strategies is a future research work worthy of investigation to improve the solution quality and computational effort of GAP of BinYaprak.

6. REFERENCES

Aksakalli, V., Alkaya, A. F., & Algin, R. (2020). Hybridization of Migrating Birds Optimization with Simulated Annealing.

Baykasoglu, A., Ozbakir, L., & Tapkan, P. (2007). Artificial bee colony algorithm and Its application to generalized assignment problem. Swarm Intelligence: Focus on Ant and particle swarm optimization, 1.

Burkard, R. E., Cela, E., Pardalos, P. M., & Pitsoulis, L. S. (1998). The quadratic assignment problem. In Handbook of combinatorial optimization (pp. 1713-1809). Springer, Boston, MA.

Dıaz, J. A., & Fernández, E. (2001). A tabu search heuristic for the generalized assignment problem. European Journal of Operational Research, 132(1), 22-38.

Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*, *217*, 65-77.

Duman, E., & Elikucuk, I. (2013, June). Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization. In *International Work-Conference on Artificial Neural Networks* (pp. 62-71). Springer, Berlin, Heidelberg.

Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. Journal of global optimization, 6(2), 109-133.

Ganesh, K., Mohapatra, S., Malairajan, R. A., & Punniyamoorthy, M. (2015). Resource Allocation Problems in Supply Chains. Emerald Group Publishing.

Gendreau, M., & Potvin, J. Y. (Eds.). (2010). Handbook of metaheuristics (Vol. 2, p. 9). New York: Springer.

Geunes, J. (2014). Operations Planning: Mixed Integer Optimization Models (Vol. 11). CRC Press.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. Decision sciences, 8(1), 156-166.

Gonzalez, T. F. (Ed.). (2018). Handbook of Approximation Algorithms and Metaheuristics: Contemporary and Emerging Applications, Volume 2. CRC Press.

Kaviani, M., Abbasi, M., Rahpeyma, B., & Yusefi, M. (2014). A hybrid tabu search-simulated annealing method to solve quadratic assignment problem. Decision Science Letters, 3(3), 391-396.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2), 83-97.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. Science 220(4598), 671–680.

Kundakcioglu, O. E., & Alizamir, S. (2009). Generalized Assignment Problem.

Küçükoğlu, İ., Dewil, R., & Cattrysse, D. (2019). Hybrid simulated annealing and tabu search method for the electric travelling salesman problem with time windows and mixed charging rates. Expert Systems with Applications, 134, 279-303.

Lalami, M. E., Elkihel, M., El Baz, D., & Boyer, V. (2012). A procedure-based heuristic for 0-1 Multiple Knapsack Problems. International Journal of Mathematics in Operational Research, 4(3), 214-224.

Lalla-Ruiz, E., Armas, J. D., Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2017). Multi-leader migrating birds optimisation: a novel nature-inspired metaheuristic for combinatorial problems. *International Journal of Bio-Inspired Computation*, *10*(2), 89-98.

Martello, S. and Toth, P., An algorithm for the generalized assignment problem, In Proceedings of IFORS International Conference on Operational Research, Brans, J.P., Ed., North-Holland, Amsterdam, the Netherlands 1981, p. 589.

Martello, S., & Toth, P. (1981). Heuristic algorithms for the multiple knapsack problem. Computing, 27(2), 93-112.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. Journal of Chemical Physics 21(6), 1087–1092.

Osman, I. H. (1995). Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. Operations-Research-Spektrum, 17(4), 211-225.

Öncan, T. (2007). A survey of the generalized assignment problem and its applications. INFOR: Information Systems and Operational Research, 45(3), 123-141.

Öz, D., & Öz, I. (2021). Scalable parallel implementation of migrating birds optimization for the multi-objective task allocation problem. *The Journal of Supercomputing*, *77*(3), 2689-2712.

Resende, M. G., Martí, R., & Panos, P. (2017). Handbook of heuristics.

Sioud, A., & Gagné, C. (2018). Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times. *European Journal of Operational Research*, *264*(1), 66-73.

Soto, R., Crawford, B., Almonacid, B., & Paredes, F. (2015, October). A migrating birds optimization algorithm for machine-part cell formation problems. In *Mexican International Conference on Artificial Intelligence* (pp. 270-281). Springer, Cham.

Tongur, V., Ertunc, E., & Uyan, M. (2020). Use of the Migrating Birds Optimization (MBO) Algorithm in solving land distribution problem. *Land Use Policy*, *94*, 104550.

Toth, P., & Martello, S. (1990). Knapsack problems: Algorithms and computer implementations (pp. 14-15). New York: Wiley.

Ulker, E., & Tongur, V. (2017). Migrating birds optimization (MBO) algorithm to solve knapsack problem. *Procedia computer science*, *111*, 71-76.

Wu, T. H., Yeh, J. Y., & Syau, Y. R. (2004). A tabu search approach to the generalized assignment problem. Journal of the Chinese Institute of Industrial Engineers, 21(3), 301-311.

APPENDIX A. Sector, Expertise and Experience Lists.

Sectors List	Profession/Expertise List	Experience List
Advertising	Artificial Intelligence	Campus Life
Architecture	Business Analytics	Cash Management
Art	Business Strategy	Choice of Profession
Aviation	Coding- Python	Entrepreneurship
Banking	Coding-C++	Erasmus Exchange Program
Computer/Electronics	Coding-Java	Erasmus Internship
Construction	Computer Engineering	Global Career
Consultancy	Data Mining	Graduate Study
Defense Industry	Data Science	Internship
Design	Design	Interview Tips
Education	Digital Marketing	Leadership
Energy/Mining	E-Commerce	Mentorship
Entertainment	Electrical Electronics Engineering	Motivation
Finance	Entrepreneurship	Networking
Fine Arts	Google Analytics	Phd Abroad
Hardware&Computer Networks	Human Resources	Social Entrepreneurship
Service Industry	Industrial Engineering	Social Responsibility Projects
Information Technologies	Machine Learning	Stress Management
Insurance Business	Management	Study Abroad
Engineering	Marketing	Sustainability
Law	Media	Sustainable Development Goals
Leisure and Travel	Operations Management	Teaching
Logistics	Product Management	University Life
Manufacturing	Project Management	University Preferences
Medicine and Wellness	R&D	Other
Media	Resource Management	
Pharmacy	Sales Department	
Publishing	Social Media	
Reail	Social Service	
Real Estate Business	Supply Chain Management	
Software Development	Tecnology and Digital	
Telecommunications	Virtual Reality	
Textile	Web Designer	
Tourism	Other	
Other		

VITA

Merve Özer received her bachelor's degree in International Trade and Management from Istanbul Şehir University, and graduated as a highest-ranking student. She has experience of studying and working at different institutions. She attended the Erasmus+ Exchange program and spent a semester in the UK to take management classes at Keele University, and then she participated in the Erasmus+ Internship program and worked as an academic coordinator at EF International Language School in Brighton / UK. After graduation, she joined the graduate program at Özyeğin University as a MSc student of Industrial Engineering under the supervision of Professor Ekrem Duman in 2020. Her research interests are operations management, supply chain management, metaheuristic approaches and optimization.