A Practical Approach Towards Multidimensional Knapsack Problem (MKP) Using Genetic Algorithm

Muhammad Irfan Leghari[1], Aftab Ahmed Shaikh[2]

***Abstract* Knapsack is a combinational optimization problem which is based on the selection of the best possible collection of values and weights capacity.The Selection of values varies and depends on the social and environmental situations. The research work is focused on the practical approach of the Multidimensional Knapsack problem (MKP). This paper comprises an introductory part of Genetic Algorithm and understanding towards Knapsack problem. The program is developed in Python programming language using the pyEasyGa libraries. The MKP program Genetic Algorithm class is initialized within a defined population size. Fitness function is defined to evaluate best solutions in the shape of best values to maximize the beneficial point.**

Keywords: Multidimensional Knapsack Problem (MKP); Genetic Algorithm; Python, Fitness Function

I INTRODUCTION

A) Genetic Algorithm

Genetic algorithm (GA) is a dynamic and heuristic search algorithm use living organisms adopted methods. The GA works on population and coding parameters which are required for solution. GA is evolved over generations and the consequent population comes by type of different data in shape of binary numbers, numbers, characters called a chromosome. Group of chromosomes is called population. Generation of initial population is arbitrarily. Chromosomes own a fitness-value. Fitness value is denoted by mathematical expression and determines the probability survival of each particular chromosome in the next generation [1]. The fittest chromosomes are survivors and lead to evolution process. The survival chances for organism variations are prettylean. Therefore, evolution is a process of natural selection [2]. Genetic algorithm, the population of various chromosomes with random genes collection is followed as per steps of Figure-1.

B) Steps of Genetic Algorithm

Fitness or objective function is an initial step of genetic algorithm solutions. Fitness function is used to evaluate the possible solutions of each chromosomes space. The raw output obtained from the fitness function is used for selection function [1]. After the fitness is evaluated the fittest chromosomes are selected with preferred scored. Chromosomes that have high fitness value, are reproduced in general [3]. Three operators of genetic algorithm i) crossover ii) mutation and iii) off-spring are applied with chromosomes with fittest values selected. Crossover applies with single point, two-point and uniform crossover selections to produce off-spring and new selections. Mutation through swapping of genes. Mutation operator applies on single individual of population to swap values [3].

Table-1: Genetic Algorithm

Steps in Genetic Algorithm
1st: Generationof an initial population of chro- .mosomes
2nd: Evaluating the suitability of each chromo- .some (individual) that forms the population
.3rd: Selecting the chromosomes
.4th: Producing offspring
.5th: Mutation ofrandom Genes
6th: Repeat (steps 3-5) tillgeneration of new .population
.7th: End of algorithm on best solution

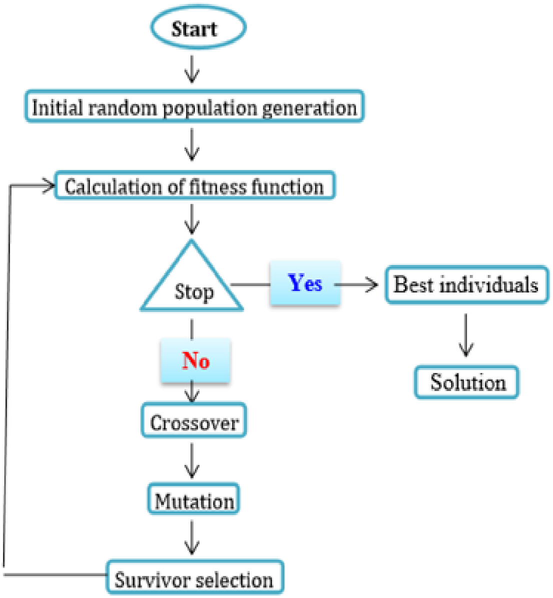


Figure-1: Genetic algorithm flowchart

1 Department of Computer Science, Sindh Madressatul Islam University,
2 Professor, Faculty of Information Technology, Sindh Madressatul Islam University

C) Knapsack Problem

Knapsack is a combinational optimization problem based on selection of the best possible collection of values within defined weight capacity. The concept of Knapsack problem is to fill the survival bag with beneficial stuff out of given list. Each item has particular benefit value and weight. The selection process follows the objective to maximize the beneficial values with selection of most useful surviving items. The Multidimensional knapsack is a most-studied problem of applied mathematics, powerfully NP-hard combinational-problem occur in different application. Knapsack problem is used to get best solution amongst the given criteria. Every time the variable fulfills different values until getting satisfied solution using the Genetic Algorithm [4]. Integer programmodel for multidimensional knapsack problem illustrated in figure-2. We have items-n and resources-m. Items generate profitsp(j) and resources have a capacity of c(i). Each item-j consumes an amount w(ij) from each resource i. The goal is to select a subset of items that maximizes total profit and not exceed resource capacities [5].

$$\textit{minimize} = \quad \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\textit{subject to:} \quad \sum_{j=1}^{n} w_{ij} x_j \le c_i \qquad i \in \{1 \dots m\} \tag{2}$$

$$x_j \in \{0,1\} \qquad j \in \{1 \dots n\} \tag{3}$$

Figure-2: Integer program model

II. RESEARCH BACKGROUND

In 2020, Dr. Manish Saraswati worked on experiment GAs meta-algorithms which applied for obtaining solutions of optimizing problems. He discussed GA application to single-instance optimization of Knapsackproblem. His experiment showed that genetic algorithm could be applied to produce higher-benefit [1].

In 2019, Lai, X., Hao presented two-stage solution on tabu search for the “multidemand multidimensional knapsack problem”. They investigated how this search could be could be used to solve this computational problem. And they proposed two-stage search algorithm, first stage proposed to locate hyperplan within whole search. Second stage finds improved solutions with reduced subspace. The experiment used in literature shows algorithm satisfactory with results [6].

In 2018, Abdellah Rezoug and Mohamed BaderEDin, presents heuristic approach ‘Guided genetic Algorithm’ for solving Multidimensional Knapsack problem. A bi-way algorithm of data pre analysis and GA. Pre analysis performed with efficiency based-method for get information. Which is integrated in GA in two stage, to generate initial population and evaluate offspring. GGA examined on major parameters turned. Impact of GGA checked with t-test [7].

In 2015, Bernhard Lienland and Li Zeng took multiple characteristics of GAs and approaches compared in multiple studies with unequal conditions. Literature reviewed 11 multiple Gas with the detailed information and performance. And the authors of review authorized the fitnessvalues. Essential computation duration in variable problem-type and environment. The outcomes showed genetic algorithm superiority [8].

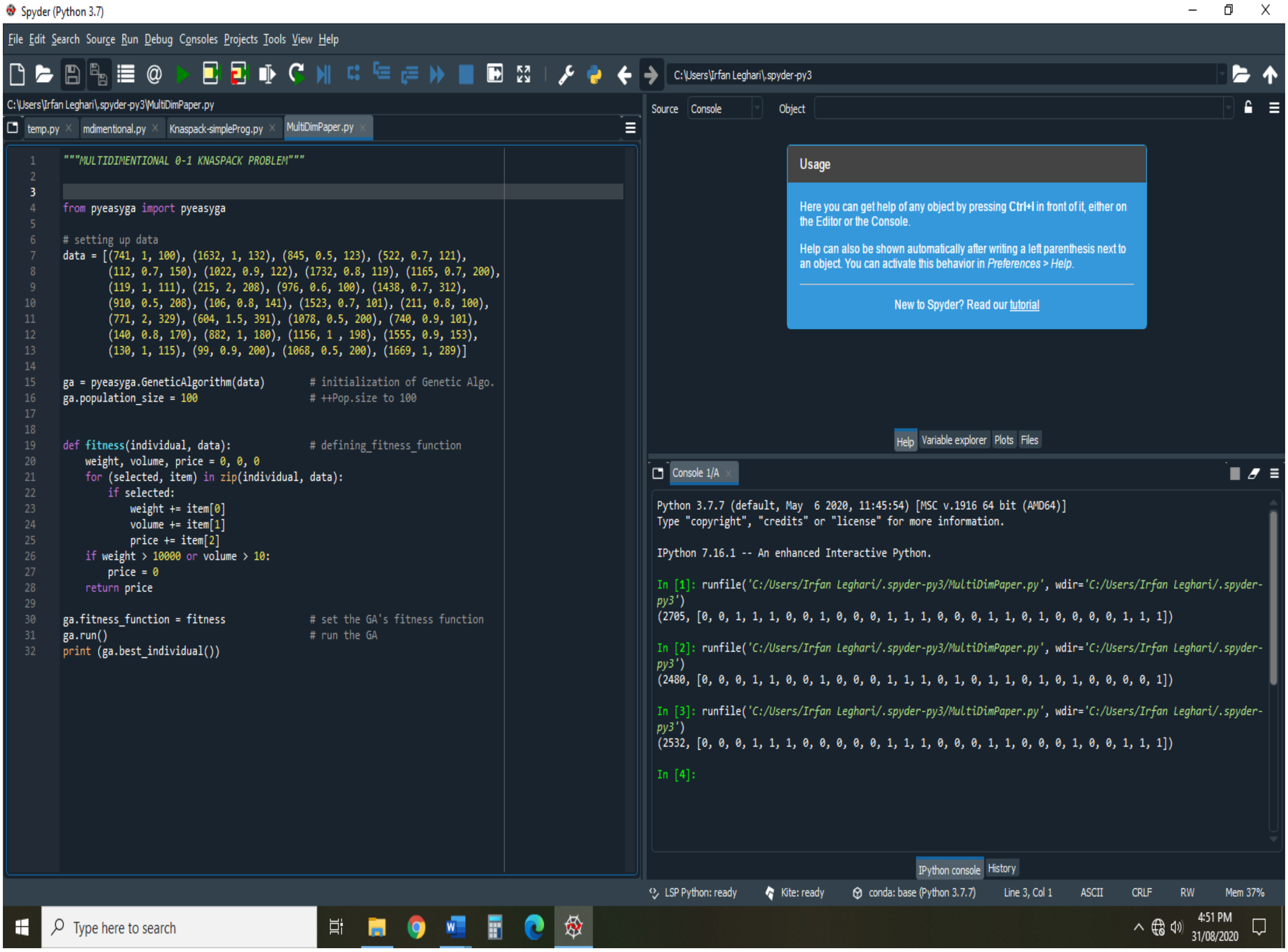
In 2013, Berberler, M. E., Guler, proposed an algorithm wherein, initial-population is not arbitrarily generated, accordingly, the solutions space required to be scanned competently. C language is opted for writing algorithm which generates optimal solutions for all instances [9].

In 2011, M. JalaliVarnamkhasti, reviewed literature of popular algorithms suitable for solving Multidimensional Knapsack Algorithm. Few algorithms exact, heuristics and metaheuristic selected in superior category. Suggested applications which includes many applicable problems from different area, like cargo loading, cutting stock, bin-packing, financial and other management [10].

III. IMPLEMENTATION WITH PYTHON

The section deals with programming solution of the multidimensional knapsack problem. The objective of the harder D-dimentional Knapsack problem is to maximize the total values of surviving items in multidimensional knapsack to avoid exceeding weight sum. 0-1 knapsack problem limits the items to 0-1, where items with weight wi and value vi. Selection of values varies and depends on the social and environmental situations. Multidimensional Knapsack problem programmed and compiled in Spyder IDE-3.7 version. Easy to used steps implemented genetic algorithm with library of pyEasyGa. Python easy genetic algorithm is a standard library in Python. Which providing an easy and simple interface. Every time the variable fulfills different values until getting satisfied solution using the Genetic Algorithm. Pyeasyga libraries satisfies required weight and volume constraints.

Multidimensional knapsack problem consists over 28 items for packing container with volume 10 to accommodate a maximum weight of 10000 in some unit. Items with certain weight, volume and price combined. The assignment is to packing in container the several items in following criteria of the container that maximum volume of items or items maximum weight may not be surpassed at the time maximizing the total price of the items packed. Given are the item’s



Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

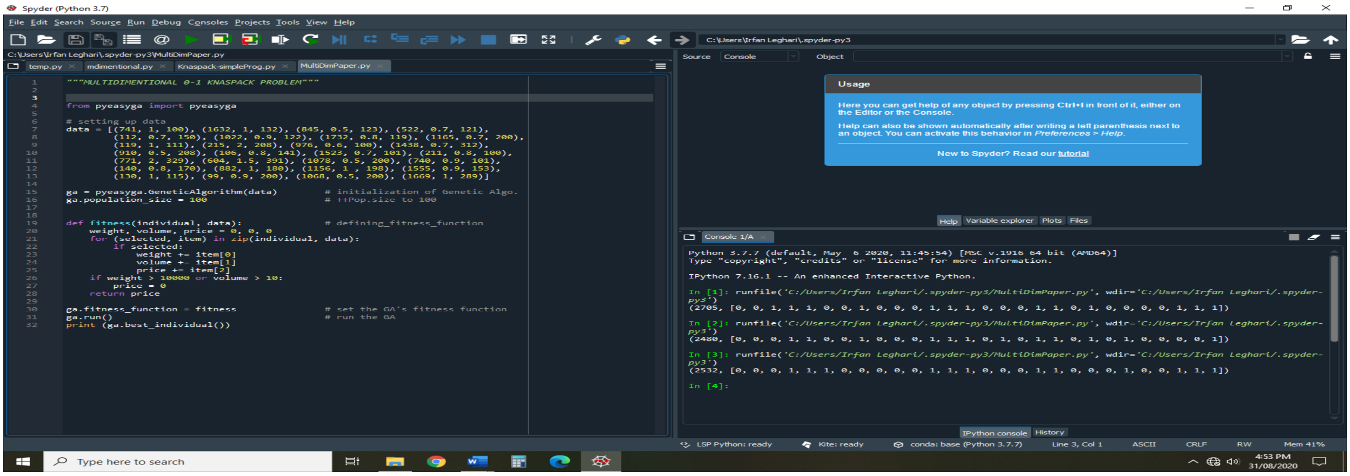
New to Spyder? Read our tutorial

```python
"""MULTIDIMENTIONAL 0-1 KNASPACK PROBLEM"""


from pyeasyga import pyeasyga

# setting up data
data = [(741, 1, 100), (1632, 1, 132), (845, 0.5, 123), (522, 0.7, 121),
        (112, 0.7, 150), (1022, 0.9, 122), (1732, 0.8, 119), (1165, 0.7, 200),
        (119, 1, 111), (215, 2, 208), (976, 0.6, 100), (1438, 0.7, 312),
        (910, 0.5, 208), (106, 0.8, 141), (1523, 0.7, 101), (211, 0.8, 100),
        (771, 2, 329), (604, 1.5, 391), (1078, 0.5, 200), (740, 0.9, 101),
        (140, 0.8, 170), (882, 1, 180), (1156, 1 , 198), (1555, 0.9, 153),
        (130, 1, 115), (99, 0.9, 200), (1068, 0.5, 200), (1669, 1, 289)]

ga = pyeasyga.GeneticAlgorithm(data)        # initialization of Genetic Algo.
ga.population_size = 100                    # ++Pop.size to 100


def fitness(individual, data):              # defining_fitness_function
    weight, volume, price = 0, 0, 0
    for (selected, item) in zip(individual, data):
        if selected:
            weight += item[0]
            volume += item[1]
            price += item[2]
    if weight > 10000 or volume > 10:
        price = 0
    return price

ga.fitness_function = fitness               # set the GA's fitness function
ga.run()                                    # run the GA
print (ga.best_individual())
```

```
Python 3.7.7 (default, May  6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Irfan Leghari/.spyder-py3/MultiDimPaper.py', wdir='C:/Users/Irfan Leghari/.spyder-
py3')
(2705, [0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1])

In [2]: runfile('C:/Users/Irfan Leghari/.spyder-py3/MultiDimPaper.py', wdir='C:/Users/Irfan Leghari/.spyder-
py3')
(2480, [0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1])

In [3]: runfile('C:/Users/Irfan Leghari/.spyder-py3/MultiDimPaper.py', wdir='C:/Users/Irfan Leghari/.spyder-
py3')
(2532, [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1])

In [4]:
```

Figure-3: MKP execution

weights of the same unit of the container are 741, 1632, 845, 522, 112, 1022, 1732, 1165, 119, 215, 976, 1438, 910, 106, 1523, 211, 771, 604, 1078, 740, 140, 882, 1156, 1555, 130, 99, 1068, 1669. Given are volumes of items in the same unit as that of the container are 1, 1, 0.5, 0.7, 0.7, 0.9, 0.8, 0.7, 1, 2, 0.6, 0.7, 0.5, 0.8, 0.7, 0.8, 2, 1.5, 0.5, 0.9, 0.8, 1, 1, 0.9, 1,



Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```python
"""MULTIDIMENTIONAL 0-1 KNASPACK PROBLEM"""


from pyeasyga import pyeasyga

# setting up data
data = [(741, 1, 100), (1632, 1, 132), (845, 0.5, 123), (522, 0.7, 121),
        (112, 0.7, 150), (1022, 0.9, 122), (1732, 0.8, 119), (1165, 0.7, 200),
        (119, 1, 111), (215, 2, 208), (976, 0.6, 100), (1438, 0.7, 312),
        (910, 0.5, 208), (106, 0.8, 141), (1523, 0.7, 101), (211, 0.8, 100),
        (771, 2, 329), (604, 1.5, 391), (1078, 0.5, 200), (740, 0.9, 101),
        (140, 0.8, 170), (882, 1, 180), (1156, 1 , 198), (1555, 0.9, 153),
        (130, 1, 115), (99, 0.9, 200), (1068, 0.5, 200), (1669, 1, 289)]

ga = pyeasyga.GeneticAlgorithm(data)        # initialization of Genetic Algo.
ga.population_size = 100                    # ++Pop.size to 100


def fitness(individual, data):              # defining_fitness_function
    weight, volume, price = 0, 0, 0
    for (selected, item) in zip(individual, data):
        if selected:
            weight += item[0]
            volume += item[1]
            price += item[2]
    if weight > 10000 or volume > 10:
        price = 0
    return price

ga.fitness_function = fitness               # set the GA's fitness function
ga.run()                                    # run the GA
print (ga.best_individual())
```

```
Python 3.7.7 (default, May  6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Irfan Leghari/.spyder-py3/MultiDimPaper.py', wdir='C:/Users/Irfan Leghari/.spyder-
py3')
(2705, [0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1])

In [2]: runfile('C:/Users/Irfan Leghari/.spyder-py3/MultiDimPaper.py', wdir='C:/Users/Irfan Leghari/.spyder-
py3')
(2480, [0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1])

In [3]: runfile('C:/Users/Irfan Leghari/.spyder-py3/MultiDimPaper.py', wdir='C:/Users/Irfan Leghari/.spyder-
py3')
(2532, [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1])

In [4]:
```

Figure-4: MKP results

0.9, 0.5, 1. Finally, the given prices of the items are 100, 132, 123, 121, 150, 122, 119, 200, 111, 208, 100, 312, 208, 141, 101, 100, 329, 391, 200, 101, 170, 180, 198, 153, 115, 200, 200, 289.

4. RESULTS

The selection of 28 items 10 volume to adjust maximum weight of 10000 satisfies the required constraints of volume and maximum weight. The execution of program returns different total values 2705, as illustrated in Figure-4. The central code functionality and description of main code is given below in each line and meaning in opposite side, refer Table-2. Multidimensional Knapsack problem for GA, the different output series that the indicated selection of items satisfies the required weight and volume constraints.

Table-2: Coding description

from pyeasyga import pyeasyga *[…………] = data* *ga = pyeasyga.Geneti(cAlgorithm(data* *ga.population_size = 100* *def fitness(individual, :(data* *ga.fitness_function = fitness* *()ga.run* *print (ga.best_individu(()al*	Importing module Setting-up data Initialization of GeneticAlgorithm_class Increase population size to 100 Defining fitness function Set the GA's fitness function Run the Genetic algorithm Print for best solutions

5.CONCLUSION

Multidimensional Knapsack Problem is practically implemented in this paper. We start studying genetic algorithm basics and algorithm steps and flow chart. And continued towards generic algorithm operators and its types. Following introduction and background study of Multidimensional Knapsack Problem we focused on functionality towards Multidimensional Knapsack Problem (MKP). The practical approach of Multidimensional Knapsack problem programmed and complied in compiled in Spyder IDE-3.7 version. The series of results of various indicate selection of items satisfies the required weight and volume, which shows in results that 28 items 10 volume to adjust maximum weight of 10000 satisfies the required constraints of volume and maximum weight.

REFERENCES:

1. Uslu, F. S. (2015, May). Solving Knapsack Problem with Genetic Algorithm. In 2015 23nd Signal Processing and Communications Applications Conference (SIU) (pp. 1062-1065). IEEE.
2. Herath, A. K. (2017). Genetic Algorithm for University Course Timetabling Problem (Doctoral dissertation, University of Mississippi).
3. Kramer, O. (2017). Genetic algorithms. In Genetic algorithm essentials (pp. 11-19). Springer, Cham.
4. Gupta, I. K., Choubey, A., & Choubey, S. (2017, July). Clustered genetic algorithm to solve multidimensional knapsack problem. In 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.
5. Puchinger, J., Raidl, G. R., &Pferschy, U. (2015). The multidimensional knapsack problem: Structure and algorithms. INFORMS Journal on Computing, 22(2), 250-265.
6. Lai, X., Hao & Yue, D. (2019). "Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem". European Journal of Operational Research, 274(1), 35-48.
7. Rezoug, Bader-El-Den, &Boughaci, (2018). "Guided genetic algorithm for the multidimensional knapsack problem". Memetic Computing, 10(1), 29-42.
8. Lienland& Zeng (2015). "A review and comparison of genetic algorithms for the 0-1 multidimensional knapsack problem". International Journal of Operations Research and Information Systems (IJORIS), 6(2), 21-31.

9 Berberler, Guler, A., & Nurıyev, (2013), "A genetic algorithm to solve the multi-dimensional knapsack problem", Mathematical and Computational Applications, 18(3), 486-494.

10, Varnamkhasti (2012), "Overview of the algorithms for solving the multi-dimensional knapsack problems". Advanced Studies in Biology, 4(1), 37-47.