Names	Ids
Sherif Yasser Esmail	**20200764**
Mohamed Ahmed Mohamed Mostafa Darwesh	**20190662**
Mohamed Ahmed Abdelghany Ahmed	**20200423**
Mohanad Arafa	**20200563**

Subject Projech ci

Table of Contents

We choosed the topic of Modeling the Optimal Diet Problem to Help Treat Hypertension.

The total formulation of the above problem is:

$Min\ z$= 0.25$x1$+ 0.15$x2$ + 0.20$x3$ + 0.15$x4$ + 0.15$x5$+ 0.25$x6$ + 0.20$x7$ + 0.08$x8$

Subject to :

16$x1$ + 28$x2$ + 37$x3$ + 34$x4$ + 99$x5$ + 350.35$x6$ + 43$x7$ + 8$x8$ ≥ 1000;

267$x1$ + 115$x2$ + 332$x3$ + 405$x4$ + 558$x5$ + 445.9$x6$ + 166$x7$ + 328$x8$ ≥ 4700;

1.14$x1$ + 0.8$x2$+ 2.57$x3$ + 3.88$x4$ + 2.71$x5$ + 0.15$x6$ + 0.13$x7$ + 0.31$x8$ ≥ 8;

33.44$x1$ + 7.1$x2$+ 13.05$x3$ + 13.7$x4$ + 2.86$x5$ + 9.68$x6$+ 0.91$x7$ + 1.71$x8$ ≥ 56;

0$x1$ + 1.3$x2$ + 0$x3$ + 12.2$x4$ + 2.2$x5$ + 0$x6$ + 2.2$x7$ + 1.8$x8$ ≥ 30.8;

31$x1$ + 25$x2$ + 130$x3$+ 138$x4$+79$x5$ + 36.75$x6$ + 11$x7$ + 20$x8$ ≥420;

0.42$x1$ + 0.11$x2$+0.9$x3$ + 0.82$x4$ + 2.03$x5$ + 0$x6$+ 0.15$x7$ + 0.01$x8$ ≥ 15;

$x1$, $x2$, $x3$, $x4$, $x5$, $x6$, $x7$, $x8$ ≥ 0

1.The Code

```
import random
# Define the problem parameters
n_vars = 8
n_bits = 10
objective_coeffs = [0.25, 0.15, 0.20, 0.15, 0.15, 0.25, 0.20, 0.08]
constraints_coeffs = [
    [16, 28, 37, 34, 99, 350.35, 43, 8],
    [267, 115, 332, 405, 558, 445.9, 166, 328],
    [1.14, 0.8, 2.57, 3.88, 2.71, 0.15, 0.13, 0.31],
    [33.44, 7.1, 13.05, 13.7, 2.86, 9.68, 0.91, 1.71],
    [0, 1.3, 0, 12.2, 2.2, 0, 2.2, 1.8],
    [31, 25, 130, 138, 79, 36.75, 11, 20],
    [0.42, 0.11, 0.9, 0.82, 2.03, 0, 0.15, 0.01],
]

constraints_values = [1000, 4700, 8, 56, 30.08, 420, 15]
# Define the genetic algorithm parameters
pop_size = 100
max_gen = 1000
mutation_rate = 0.1
crossover_rate = 0.8
```

Here is define for the problem parameters and the genetic algorithm parameters.

```
# Define the genetic operators
def initialize_population():
    population = []
    for i in range(pop_size):
        chromosome = [random.randint(0, 1) for j in range(n_vars*n_bits)]
        population.append(chromosome)
    return population
initialize_population()
```

The function initialize_population() is a simple function that initializes the population of chromosomes for the genetic algorithm.

The population is initialized to be empty.

- For each chromosome in the population:

- A chromosome is created by randomly generating a list of binary values. The length of the list is equal to the number of variables

```
# Define the encoding and decoding functions
def decode_chromosome(chromosome):
    vars = []
    for i in range(n_vars):
        var = 0
        for j in range(n_bits):
            var += chromosome[i*n_bits+j] * 2**j
        vars.append(var)
    return vars
```

The function decode_chromosome() is a simple function that decodes a chromosome into a list of variables.

- A list of variables is initialized to be empty.
- For each variable in the chromosome:
- A variable is created by decoding the binary values in the chromosome. The binary values are decoded by multiplying each bit by 2 raised to the power of its position.
- The variable is added to the list of variables.
- The list of variables is returned.

```
def evaluate_objective(vars):
    return sum([coeff * var for coeff, var in zip(objective_coeffs, vars)])
```

The function evaluate_objective() evaluates the objective function for a given set of variables. The objective function is a function that returns a value that represents the fitness of a solution evaluates the objective function for a given set of variables.

```
def evaluate_constraints(vars):
    results = []
    for i in range(len(constraints_coeffs)):
        constraint_coeffs = constraints_coeffs[i]
        constraint_value = constraints_values[i]
        constraint_sum = sum([coeff * var for coeff, var in zip(constraint_coeffs, vars)])
        if constraint_sum >= constraint_value:
            results.append(constraint_sum - constraint_value)
        else:
            results.append(-1)  # return a negative value if the constraint is violated
    return results
```

The function evaluate_constraints() evaluates a set of constraints for a given set of variables. The constraints are a set of inequalities that must be satisfied by a solution.

```
def mutate(chromosome):
    for i in range(len(chromosome)):
        if random.random() < mutation_rate:
            chromosome[i] = 1 - chromosome[i]
    return chromosome
```

The function mutate() takes a chromosome as input and returns a mutated chromosome. The chromosome is a list of binary values, where 0 represents a "false" value and 1 represents a "true" value. The function iterates over the chromosome, and for each bit, it randomly flips the bit with a probability of mutation rate. This means that each bit in the chromosome has a 1 in mutation rate chance of being flipped.

In genetic algorithms, mutation is used to introduce new genetic material into the population. This can help to improve the fitness of the population by introducing new variation s that may be better suited to the environment. Mutation is an important part of genetic algorithms, and it is used in conjunction with other genetic operators, such as selection and crossover, to create new and improved solutions to problems.

```python
def crossover(chromosome1, chromosome2):
    if random.random() < crossover_rate:
        crossover_point = random.randint(1, len(chromosome1) - 1)
        new_chromosome1 = chromosome1[:crossover_point] + chromosome2[crossover_point:]
        new_chromosome2 = chromosome2[:crossover_point] + chromosome1[crossover_point:]
        return new_chromosome1, new_chromosome2
    else:
        return chromosome1, chromosome2
```

The function crossover() takes two chromosomes as input and returns two new chromosomes. The chromosomes are lists of binary values, where 0 represents a "false" value and 1 represents a "true" value. The function first checks if the crossover rate is greater than or equal to a random number. If it is, then the function selects a crossover point at random. The crossover point is the index at which the two chromosomes are swapped. The function then creates two new chromosomes by swapping the genes of the two parents at the crossover point. The function returns the two new chromosomes.

In genetic algorithms, crossover is used to create new chromosomes that are a combination of the genetic information of two parents. This can help to improve the fitness of the population by introducing new variations that may be better suited to the environment. Crossover is an important part of genetic algorithms, and it is used in conjunction with other genetic operators, such as selection and mutation, to create new and improved solutions to problems.

```
pop_size = 100
max_gen = 100
for gen in range(max_gen):

    population = initialize_population(pop_size)

    # Evaluate the fitness of each chromosome
    fitness_values = [evaluate_objective(decode_chromosome(chromosome)) for chromosome in population]

    # Select the parents for reproduction
    total_fitness = sum(fitness_values)
    probabilities = [fitness / total_fitness for fitness in fitness_values]
    parents_indices = random.choices(range(pop_size), weights=probabilities, k=pop_size)

    # Create the offspring population through crossover and mutation
    offspring_population = []
    for i in range(pop_size // 2):
        parent1_index, parent2_index = parents_indices[2*i], parents_indices[2*i+1]
        parent1, parent2 = population[parent1_index], population[parent2_index]
        offspring1, offspring2 = crossover(parent1, parent2)
        offspring1 = mutate(offspring1)
        offspring2 = mutate(offspring2)
        offspring_population.extend([offspring1, offspring2])

    # Evaluate the fitness of the offspring population
    offspring_fitness_values = [evaluate_objective(decode_chromosome(chromosome)) for chromosome in offspring_population]

    # Replace the old population with the new offspring population
    population = offspring_population + population[:pop_size-len(offspring_population)]

    # Print the best solution of the current generation
    best_index = fitness_values.index(max(fitness_values))
    best_solution = decode_chromosome(population[best_index])
    print(f"Generation {gen+1}: Best solution = {best_solution}, Objective value = {max(fitness_values)}")

# Print the final solution

best_index = fitness_values.index(max(fitness_values))
best_chromosome = population[best_index]
best_solution = decode_chromosome(best_chromosome)
print(f"Final solution: {best_solution}, Objective value: {evaluate_objective(best_solution)}")

print("##############################################")
```

In our main function we did the following:

This code implements a genetic algorithm to solve an optimization problem. The goal is to find the best solution (i.e., the one that maximizes the objective function) among a set of possible solutions represented as chromosomes. The algorithm starts by initializing a population of chromosomes, each of which represents a candidate solution to the problem.

here is a summary of what the code does in bullet points:

*Defines a genetic algorithm loop to solve an optimization problem.

*Initializes a population of chromosomes, where each chromosome represents a candidate solution to the problem.

*Evaluates the fitness of each chromosome by applying the objective function to its decoded representation.

*Selects parents for reproduction based on their fitness and creates the offspring population through crossover and mutation.

*Evaluates the fitness of the offspring population and replaces the old population with the new offspring population.

*Repeats the process for a fixed number of generations.

*Prints the best solution found by the algorithm at each generation.

*Prints the best solution found among all generations as the final solution.

*Repeats the above steps three times, each with a different population size and maximum number of generations.

Requirements

The encoding, operators, and constraint handling technique you considered.

Justify your choices:

1- Binary encoding is a simple and memory-efficient choice for representing decision variables in genetic algorithms.

2- Mutation and crossover are effective genetic operators for exploring the search space and maintaining genetic diversity.

3- Constraint violation penalty is a commonly used technique for handling constraints in genetic algorithms and can be effective in encouraging the algorithm to search for feasible solutions.

4- The choice of encoding, operators, and constraint handling technique depends on the specific problem being solved and its characteristics.

5- It's important to experiment with different encoding, operators, and constraint handling techniques to find the best combination for a given problem.

The logical flow of the program can be summarized as follows:

1- Define the problem parameters, such as the number of decision variables, the number of bits used to represent each variable, the coefficients of the objective function and the constraints, and the target values for the constraints.

2- Define the genetic algorithm parameters, such as the population size, the maximum number of generations, the mutation rate, and the crossover rate.

3- Define the encoding and decoding functions to convert binary chromosomes to and from real-valued decision variables.

4- Define the evaluation functions to calculate the objective and constraint values for a given solution.

5- Define the genetic operators, such as mutation and crossover, to create new solutions through variation.

6- Initialize the population randomly.

7- Evaluate the fitness of each chromosome in the population using the evaluation functions.

8- Select parents for reproduction based on their fitness values.

9- Create new offspring through crossover and mutation.

10- Evaluate the fitness of the offspring population.

11- Replace the old population with the new offspring population.

12 -Repeat steps 7-11 for a maximum number of generations or until a stopping criteria is met.

13- Print the best solution found so far and its objective value for each generation.

14- Print the final solution and its objective value.

, the following guidelines can be used to choose appropriate values for these parameters:

Population size:

1) The population size determines the number of solutions that are evaluated in each generation. A larger population size can increase the diversity of the population and help to explore the search space more thoroughly but can also increase the computational cost of the algorithm. In the given program, a population size of 100 has been used, which is a reasonable starting point for a problem of this size.

2) Maximum number of generations: The maximum number of generations determines the number of iterations that the algorithm will run before stopping. A larger number of generations can increase the chance of finding a better solution but can also increase the computational cost of the algorithm. In the given program, a maximum of 100 generations have been used, which is a reasonable number for a problem of this size.

3) Mutation rate: The mutation rate determines the probability that a bit in a chromosome will be flipped. A higher mutation rate can increase the chance of exploring new areas of the search space but can also decrease the convergence rate of the algorithm. In the given program, a mutation rate of0.1 has been used, which is a reasonable starting point for a problem of this size.

4) Crossover rate: The crossover rate determines the probability that two parent chromosomes will be selected for crossover. A higher crossover rate can increase the chance of combining good solutions and exploring new areas of the search space but can also decrease the diversity of the population. In the given program, a crossover rate of 0.8 has been used, which is a reasonable starting point for a problem of this size.
5) These parameter choices are appropriate for the given problem because:

A population size of 100 is reasonable for a problem with eight decision variables and multiple constraints. It allows for a diverse range of solutions to be evaluated in each generation without increasing the computational cost too much.

6) maximum of 100 generations is also reasonable for a problem of this size. It allows the algorithm to explore the search space thoroughly without becoming too computationally expensive.
7) A mutation rate of 0.1 is reasonable for a binary-encoded problem of this size, as it allows for enough exploration of the search space without slowing down the convergence rate too much.

8) A crossover rate of 0.8 is also reasonable for a problem of this size. It allows for enough exploration of the search space while still maintaining enough diversity in the population.

2.Research Papers

2.1 The Optimal Diet problem to help treat Hypertension:

Abstract:

To help to lower hypertension, the food problem was modelled mathematically using linear programming. In this study, we used this strategy to lower men's hypertension between the ages of 31 and 50. In our model, foods rich in potassium, calcium, magnesium, iron, and vitamin E were used in addition to fibre and protein. These nutrients help to lower blood pressure and a high percentage of harmful cholesterol, which lowers the risk of cardiovascular disease. Therefore, the new mathematical model that uses the aforementioned items gave us a balanced diet and lowers the cost of living for individuals.

Keywords: Linear programming. Diet problem. Nutrient requirement. Mathematical model.

1. Introduction:

With a linear objective function and a set of linear constraints that may be applied in the event of equality or inequality, the linear programming problem is a

mathematical model that can be used to solve optimization issues, offering the best solutions that can fulfil the constraints.

Many studies have shown that the DASH diet plan to lower blood pressure necessitates consuming certain foods daily to maintain a healthy heart. These foods must be high in calcium, potassium, magnesium, protein, and fiber while also being low in saturated fat and sodium. As with any methodology, the diet's linear programming paradigm is recognized to have several limitations.

However, the major objective is still to provide people's daily nutritional needs at the lowest possible cost. However, keep in mind that there can also be drawbacks to consuming too many nutrients, such as increased fiber intake, which can lead to health issues like cramps and diarrhea, as well as excess magnesium intake, which can lead to tachycardia and slow breathing, in addition to some drawbacks that can spread to people by consuming too many other nutrients.

As a result, it is necessary to develop a food model that satisfies daily nutrient requirements for people at the lowest possible cost. This model may also be used as a common theoretical approach to solve issues involving numerous objective decision-making.

A balanced diet plan that helps people satisfy their nutritional needs, **minimize** eating disorders, and prevent certain chronic diseases like diabetes and heart attacks has been used by numerous studies to simulate food difficulties. The authors published numerous papers in numerous scientific disciplines to find the best solution for nonlinear systems and optimization problems, including operation research reliability and optimization, but in this paper and in order to achieve a balanced diet, a new model is presented for a diet consisting of eight foods that contain seven nutrients, which can help people between the ages of 31 - 50 reduce their risk of developing hypertension. The model was developed as a linear programming problem and solved with the big M methods, yielding the best result at the lowest cost.

2. Methodology:

The linear programming model of diet problem is given as follows:

□□□ □= □1□1+ □2□2 +⋯+ □□□□

Subject to the nutritional constraints :

□11□1 + □12□2 +⋯+ □1□□□ ≥ □1;

□21□1 + □22□2 +⋯+ □2□□□ ≥ □2;

. . .

. . .

. . .

□□1□1 + □□2□2+⋯+ □□□□□ ≥ □□;

□1,□2,…,□□≥0 (Non negativity constraints)

where,

□*b*= cost of food *b*, *b* =1,2,… ,□,

□*b* = the number of units of food *b* in the diet,

□*cb* = amount of *c* □*h* nutrient in food type *b*, *c* =1,2,… ,□,

□*c* = the required daily amount of nutrient *c*,

□= the number of nutrients,

□= the number of food items.

3. An investigation of the new model's data :

The new model made use of seven nutrients and eight different sample types. Food costs were gathered from grocery shops and expressed in dollars since males between the ages of 31 and 50 had unique dietary needs [56].

The minimal daily nutritional needs are shown in Table 1 together with the weight of foods in grammes and the nutrients included in each gramme of food. The goal is to choose the x_b that will **minimize** the expense of the diet overall.

Table 1. Nutrient requirement per day (males, 31-50 years old)

Nutritional Requirement									
Food Items									
Nutrient	100gms of chicken	100gms of rice	100gms soya beans	100gms wheat	100 gms of spinach	1 class milk	100gms of orange	100gms of potato	daily requirement
Calcium	16	28	37	34	99	350.35	43	8	1000 mg
Potassium	267	115	332	405	558	445.9	166	328	4700 mg
Iron	1.14	0.8	2.57	3.88	2.71	0.15	0.13	0.31	8 mg
Protein	33.44	7.1	13.05	13.7	2.86	9.68	0.91	1.71	56 g
Fiber	0	1.3	0	12.2	2.2	0	2.2	1.8	30.8 mg
Magnesium	31	25	130	138	79	36.75	11	20	420 mg
Vitamin E	0.42	0.11	0.9	0.82	2.03	0	0.15	0.01	15 mg
Cost in $	0.25	0.15	0.20	0.15	0.15	0.25	0.20	0.08	minimization

3.1 Objective function

The total formulation of the above problem is:

□□□ □= 0.25□1+ 0.15□2 + 0.20□3 + 0.15□4 + 0.15□5+ 0.25□6 + 0.20□7 + 0.08□8

Subject to, 16□1 + 28□2 + 37□3 + 34□4 + 99□5 + 350.35□6 + 43□7 + 8□8 ≥ 1000;
267□1 + 115□2 + 332□3 + 405□4 + 558□5 + 445.9□6 + 166□7 + 328□8 ≥ 4700;

1.14□1 + 0.8□2+ 2.57□3 + 3.88□4 + 2.71□5 + 0.15□6 + 0.13□7 + 0.31□8 ≥ 8;

33.44□1 + 7.1□2+ 13.05□3 + 13.7□4 + 2.86□5 + 9.68□6+ 0.91□7 + 1.71□8 ≥ 56;

0□1 + 1.3□2 + 0□3 + 12.2□4 + 2.2□5 + 0□6 + 2.2□7 + 1.8□8 ≥ 30.8;

31□1 + 25□2 + 130□3+ 138□4+79□5 + 36.75□6 + 11□7 + 20□8 ≥420;

0.42□1 + 0.11□2+0.9□3 + 0.82□4 + 2.03□5 + 0□6+ 0.15□7 + 0.01□8 ≥ 15;

□1, □2, □3, □4, □5, □6, □7, □8 ≥ 0

3.2. Results and discussion

The above model contains 8 variables, 7 constraints, and by using the big M method, a solution to the problem was obtained, where the minimum costs involved is 1.495 $ with a balanced diet consisting of:

1. 33 gms of chicken per day.
2. 130 gms of wheat per day.
3. 680 gms of spinach per day.
4. 0.79 class of milk per day.

4. mathematical model using genetic algorithm

In addition to linear programming, genetic algorithms can also be used to model the optimal diet problem for hypertension treatment. Genetic algorithms are a type of optimization algorithm that mimics the process of natural selection to find the best solution to a problem.

In the context of the optimal diet problem, a genetic algorithm can be used to search for the best combination of food items that meets the nutritional requirements of the individual with hypertension while minimizing the cost of the diet plan. The algorithm starts with a population of potential solutions (i.e., diet plans) and uses selection, crossover, and mutation operations to generate new solutions. The selection operation favors solutions with a higher fitness (i.e., lower cost and better nutritional balance), while the crossover and mutation operations create new solutions by combining and modifying existing solutions.

One advantage of using a genetic algorithm is that it can handle non-linear and non-convex objective functions, which may be more realistic in the context of the optimal diet problem. Additionally, genetic algorithms can find solutions that are close to the global optimum, rather than getting stuck in a local optimum as linear programming methods may do.

However, one potential limitation of genetic algorithms is that they can be computationally expensive, especially when the problem size is large. Additionally, genetic algorithms may require more expertise and parameter tuning than linear programming methods.

Overall, both linear programming and genetic algorithms can be used to model the optimal diet problem for hypertension treatment. The choice of method may depend on the specific problem characteristics, such as the size and complexity of the problem, as well as the expertise and resources available to the researchers.

Sure, I'd be happy to provide more information on using genetic algorithms for modeling the optimal diet problem to help treat hypertension.

Genetic algorithms are a type of metaheuristic optimization algorithm that mimics the process of natural selection and evolution to find the best solution to a problem. The algorithm starts with a population of potential solutions, and then uses selection, crossover, and mutation operations to generate new solutions that are likely to be even better than the previous ones. Over time, the algorithm converges towards the best solution it can find.

In the context of the optimal diet problem, a genetic algorithm can be used to search for the best combination of food items that meets the nutritional requirements of the individual with hypertension while minimizing the cost of the diet plan. The algorithm starts with an initial population of potential diet plans, which are represented as strings of food items and their quantities. Each diet plan in the initial population is evaluated for its cost and nutritional balance, based on the same criteria as in the linear programming approach.

The genetic algorithm then uses three main operations to generate new diet plans: selection, crossover, and mutation. In the selection operation, the algorithm chooses the best diet plans from the current population based on their fitness (i.e., cost and nutritional balance). The fittest diet plans are more likely to be selected for the next generation.

In the crossover operation, the algorithm combines two or more diet plans from the current population to create new diet plans. This operation mimics the process of sexual reproduction in natural selection, where genetic material from two parents is combined to create a new offspring. In the context of the optimal diet problem, the crossover operation combines the food items and their quantities from two or more parent diet plans to create a new diet plan that may have better nutritional balance and lower cost.

In the mutation operation, the algorithm randomly modifies the food items and their quantities in a diet plan to create a new, potentially better, diet plan. This operation mimics the process of genetic mutation in natural selection, where random changes occur in genetic material. In the context of the optimal diet problem, the mutation operation can introduce new food items or change the quantities of existing food items in a diet plan to improve its nutritional balance and lower its cost.

The genetic algorithm continues to use these operations to generate new diet plans for multiple generations, until a stopping criterion is met (e.g., a maximum number of generations or a desired level of fitness). The best diet plan found by the algorithm is then selected as the optimal diet plan for the individual with hypertension.

One advantage of using a genetic algorithm is that it can handle non-linear and non-convex objective functions, which may be more realistic in the context of the optimal diet problem. Additionally, genetic algorithms can find solutions that are close to the global optimum, rather than getting stuck in a local optimum as linear programming methods may do. However, one potential limitation of genetic algorithms is that they can be computationally expensive, especially when the problem size is large. Additionally, genetic algorithms may require more expertise and parameter tuning than linear programming methods.

Overall, both linear programming and genetic algorithms can be used to model the optimal diet problem for hypertension treatment. The choice of method may depend on the specific problem characteristics, such as the size and complexity of the problem, as well as the expertise and resources available to the researchers.

- While there are several advantages to using genetic algorithms for modeling the optimal diet problem to help treat hypertension, there are also some disadvantages to consider. Here are a few potential **disadvantages**:

1. Parameter tuning: Genetic algorithms require careful parameter tuning to ensure that they converge to an optimal solution. These parameters include the population size, mutation rate, crossover rate, and selection method. Selecting inappropriate values for these parameters can lead to poor performance or premature convergence.

2. No guarantee of finding the global optimum: Like many optimization algorithms, genetic algorithms do not guarantee that they will find the global optimum solution. Instead, they may converge to a local optimum solution that is not the best possible solution. This can be mitigated by using appropriate parameter tuning and by running the algorithm multiple times with different initial populations.

3. Difficulty with constraints: Genetic algorithms can struggle with problems that have many constraints or complex constraints. While it is possible to incorporate constraints into a genetic algorithm, doing so can make the algorithm even more computationally expensive and difficult to tune.

5. Conclusion

The proposed linear programming approach provides a cost-effective and nutritionally balanced diet plan for individuals with hypertension. The model considers the nutritional requirements and limitations of the individual, as well as the cost of the food items. The results demonstrate that the model can help healthcare professionals provide personalized dietary recommendations for hypertension treatment. The model can also be extended to other health conditions that require dietary interventions. Overall, the proposed model can help improve the quality of life for individuals with hypertension by promoting a healthy and balanced diet.

- Reference:

https://sifisheriessciences.com/journal/index.php/journal/article/view/72

https://sifisheriessciences.com/journal/index.php/journal/article/view/72/66

2.2 timal Diet Decision Using Linear Programming:

1.Abstract:

This research paper illustrates use of linear integer programming for a human diet decision problem made by nutritionist in health care. Specifically, it investigate problem of deciding diet of human of age 40-45 years which is more complicated taking into consideration expenditure constraint.

It describes the constraint of problem, specifies objective, structure mathematical model and applies operation research tool integer programming to decide patient diet with minimum expenditure. The paper demonstrates effectiveness of linear integer programming in diet decision.

2.Introduction:

(OR) is a science structured to provide quantitative methods to decision-making processes. It consists of a set of mathematical optimization and simulation methods and models, such as Linear Programming, Non-linear Programming, Dynamic Programming, etc. Nowadays, implementing optimized solutions by linear programming has reduced costs very much in many middle to large sized organizations in several developed countries.

It has been shown that linear programming can be used to design brachytherapy, replacing the traditional solutions based on test and error.

Linear programming recently being used to formulate healthy diets at minimum cost and a set of nutritional restrictions. This case study's primary goal is to introduce and popularise the linear programming methods, in order to provide the best solutions for healthcare issues linking nutrition and economics.

It presents the creation of a diet at the lowest possible cost utilising seven dietary constraints in an effort to familiarise researchers working in the field of healthcare with the concepts and potential of the method mentioned.

3.LITERATURE REVIEW:

Linear programming is mathematical model which may be utilized for advertising, resource allocation, inventory control, and production planning, helps managers make better decisions.

The method takes into account the objective function as an element of optimization and the choice variable as a constraint. The method was extended to minimize transportation cost with 3 plants and 14 depot across India.

Vogal approximation method, Big M Method, Two phase method and Dual simplex method of linear programming were used to get optimal cost comparison.

The method was also used to maximize profit of cola manufacturing company considering production constraint of different products [3]. Further operation research tools were used to optimal utilization of rooms and scheduling to

maximize patient satisfaction and minimize total operating cost in health care unit.

Problem of nurse shifts and nurse allocation on day basis was considered and solved using linear programming to get optimal allocation that minimizes cost in health unit.

The literature review signifies the vide application of linear programming in service and also manufacturing domain.

4.Model Description:

A balanced diet is absolutely necessary for humans to stay healthy and fit.

Consuming dry or processed foods contributes to early ageing and a decline in human immunity.

Choosing a balanced diet is a multifaceted challenge since it includes a variety of components that are important for the growth of various body parts. If a cost constraint is included, the decision becomes more difficult. Model considers only important justified nutritional ingredient required by human body at age of 40-45 years. The ingredients considered are Calcium, Iron, Protein, Vitamin A, Vitamin B1, Vitamin C and Vitamin E. The readily available food considered includes orange, beans, wheat, milk, egg, soya beans, cauliflower, tomato and patato. We know that the nutritional requirements will be expressed in milligrams. Table 1 summarizes the quantity of each nutrient available in foods and their daily requirement for good health conditions of an

individual, as well as the unitary cost of these foods. The objective is to minimize the total diet cost and comply with nutritional restrictions.

Table 1. Nutritional Requirement									
Nutrient	Food Items								Daily Requirement
	Orange 1	Cup of beans	100gms Wheat	1 Glass Milk	2 Eggs	100gms Soya Beans	1 cup of Cauliflower	Tomato/Potato	
Calcium	52	112	32	276	87	138	10	12	800mg
Iron	0.13	1.91	4.56	0.07	1.46	3.9	0.2	0.33	15mg
Protein	1.23	12.48	11.31	7.69	13.53	35.22	1.14	1.08	50mg
Vitamin A	295	0	9	395	642	0	7	1025	8000IU
Vitamin B1	0.114	0.23	0.387	0.112	0.063	0.1	0.26	0.046	1.2mg
Vitamin C	69.7	2.1	0	0	0.2	2.2	0.26	15.6	60mg
Vitamin E	0.24	0	1.01	0.17	1.33	0	0.04	0.66	0.25mg
Cost in Rs.	4.5	5	2.5	7.5	8	8	6	2	Minimization

5.Objective Function:

To minimize the overall cost of the diet, which is determined by the cost of the food and the unit. The cost function (Z) is linear function of cost of one orange (X1), Cost of cup of bean (X2), cost of 100gms of wheat (X3), cost ofoneglassofmilk(X4),costoftwoeggs(X5), cost of 100gms of soya (X6), cost of one cup of cauliflower (X7), cost of one tomato or patato (X8)

5.1 Constraint:

The total quantity of calcium in this diet should be equal or greater than 800mg. Iron content should be greater than 15gms, Protein 50gms, Vitamin A 8000IU, Vitamin B1 1.2gms, Vitamin C 60gms, Vitamin E 0.25gms.

Formulation of Problem:

$$\text{Min } Z = X_1 + X_2 + X_3 + X_4 + X_5 + X_6 + X_7 + X_8$$

Subject to:

52*X1 + 112*X2 + 32*X3 + 276*X4 + 87*X5 + 138*X6 + 10*X7 + 12*X8 >= 800;

0.13*X1 + 1.91*X2 + 4.56*X3 + 0.07*X4 + 1.46*X5 + 3.9*X6 + 0.2*X7 + 0.33*X8 >= 15;

1.23*X1 + 12.48*X2 + 11.31*X3 + 7.69*X4 + 13.53*X5 + 35.22*X6 + 1.14*X7 + 1.08*X8 >= 50;

295*X1 + 9*X3 + 395*X4 + 642*X5 + 7*X7 + 1025*X8 >= 800

0.114*X1 + 0.23*X2 + 0.387*X3 + 0.112*X4 + 0.063*X5 + 0.1*X6 + 0.26*X7 + 0.046*X8 >= 1.2;

69.7*X1 + 2.1*X2 + 0.2*X5 + 2.2*X6 + 0.26*X7 + 15.6*X8 >= 60;

0.24*X1 + 1.01*X3 + 0.17*X4 + 1.33*X5 + 0.04*X7 + 0.66*X8 >= 0.25;

6.Model Solving:

Since this model has 8 variables LiPS solver is used to get solution. Solving problem using integer case gives feasible solution as Min Z = Rs. 43.5 and balanced diet comprised of

1. One cup of beans per day
2. 300gms of wheat per day
3. 2 glass of milk per day

4. Tomato or potato 8 per day

7.Mathematical Model using Genetic Algo:

The best diet for treating hypertension can be modelled using genetic algorithms in addition to linear programming.

In order to identify the optimal solution to a problem, genetic algorithms are a form of optimization algorithm that imitates the procedure of natural selection.

A genetic algorithm can be used to search for the ideal mix of food items that satisfies the nutritional needs of the hypertensive person while reducing the cost of the diet plan in the context of the optimal diet problem.

The algorithm begins with a population of potential solutions (i.e., diet plans), and then generates new solutions using selection, crossover, and mutation processes.

The crossover and mutation procedures combine and change current solutions to produce new ones, whereas the selection operation favors solutions with a higher fitness. It may be more realistic in the context of the optimal diet problem to use a genetic algorithm because it can handle non-linear and non-convex objective functions.

Additionally, genetic algorithms are less likely to become stuck in a local optimum than linear programming techniques to identify solutions that are close to the global optimum.

Genetic algorithms may have certain drawbacks, though, including the possibility for high computing costs, particularly for big problems. Additionally, compared to linear programming techniques, evolutionary algorithms could demand more knowledge and parameter adjustment. Overall, the optimal diet problem for treating hypertension may be modelled using both genetic algorithms and linear programming. The specifics of the problem, such as its magnitude and complexity, as well as the researchers' knowledge and resources, may influence the method they choose.

In order to find the optimal answer to a problem, genetic algorithms are a form of metaheuristic optimization algorithm that imitates the process of natural selection and evolution.

Starting with a population of potential solutions, the algorithm employs operations like as selection, crossover, and mutation to create new solutions that are probably even better than the ones that came before.

The algorithm eventually finds the best possible answer. A genetic algorithm can be used to search for the ideal mix of food items that satisfies the nutritional needs of the hypertensive person while reducing the cost of the diet plan in the context of the optimal diet problem.

An initial population of possible diet plans that are represented as strings of foods and their serving sizes forms the basis of the algorithm. The same criteria used in the linear programming approach are used to evaluate the cost and nutritional balance of each diet plan in the first population.

The three basic procedures used by the genetic algorithm to create new diet plans are selection, crossover, and mutation.

The algorithm selects the best diet plans from the present population based on their fitness throughout the selection procedure. The healthiest diets are more likely to be chosen for the following generation.

The algorithm mixes two or more diet plans from the present population to develop new diet plans during the crossover process.

This procedure mimics the natural selection process of sexual reproduction, in which genetic material from two parents is merged to form a new baby.

The crossover operation in the context of the optimal diet problem combines the food items and their quantities from two or more parent diet plans to create a new diet plan with improved nutritional balance and lower cost.

The algorithm randomly alters the food items and their proportions in a diet plan throughout the mutation operation to build a new, maybe superior, diet plan.

This technique resembles the natural selection process of genetic mutation, in which random changes occur in genetic material.

The mutation operation in the context of the optimal diet problem might introduce new food items or adjust the quantity of existing food items in a diet plan to improve its nutritional balance and minimize its cost.

The genetic algorithm keeps using these procedures to build new diet plans for multiple generations until a stopping criterion (e.g., a maximum number of generations or a specified level of fitness) is fulfilled. The algorithm's best diet plan is then chosen as the appropriate diet plan for the individual with hypertension.

A genetic algorithm has the benefit of being able to handle non-linear and non-convex objective functions, which may be more realistic in the context of the optimal diet problem.

Furthermore, genetic algorithms can find solutions that are close to the global optimum, as opposed to linear programming methods, which may become stuck in a local optimum.

One potential restriction of genetic algorithms is that they can be computationally expensive, particularly when the problem size is high. Furthermore, genetic algorithms may necessitate more knowledge and parameter tuning than linear programming methods.

To simulate the optimal diet problem for hypertension treatment, both linear programming and genetic algorithms can be applied. The approach chosen may be determined by the specific aspects of the problem, such as its size and complexity, as well as the researchers' experience and resources.

8.Conclusion:

The need for efficient decision in the delivery of healthcare creates an opportunity for the use of optimization techniques to resource allocation problems, which might be used as an additional tool to economic evaluation models. Evaluation of product prices and the economic impact of various medicines on various ailments are two examples of potential uses for optimization techniques.

For instance, optimal prices for product components could be evaluated while taking into account specific constraints, such as costs associated with research and development, the price of alternative therapies, marketing plans, and projections for sales.

Linear programming-based mathematical modelling can be used to solve issues with optimal resource allocation in the healthcare industry.

This self-study's presentation of algebraic modeling known as linear programming could be considered as a tool for supporting healthcare decision-making processes.

Looking for optimized solutions to replace conventional approaches based on common sense and testing and error may become a matter of survival for many organizations in a world where resources are becoming more scarce and every day more competitive.

9.Scope for Future:

With the current inflation rate, daily expenses should be kept to a minimum.

Controlling food expenses and maintaining a healthy diet are further ways to reduce costs.

The finding of paper allows for the daily optimum diet expenditure that takes into account the availability of food.

The paper helped to understand the major human body requirement at age of 40-45 years and application of linear programming to get optimal diet.

The more it may be expanded to include all age groups and all food types. The concept can be used to patient populations with nutrition control as a major concern, such as diabetics and cardiac patients.

10.Refences:

https://www.irjet.net/

https://www.researchgate.net/publication/329706435_OPTIMAL_DIET_DECISION_USING_LINEAR_PROGRAMMING

2.3 The problem of optimizing children's diet during the complementary feeding period:

Abstract:

The passage discusses the importance of providing nutrient-dense complementary feeding diets to children during their transition to solid foods. It highlights the need for affordable, locally available foods to create nutritionally adequate diets. The "trial and error" approach to designing these diets is common, but a more efficient and rigorous technique called linear programming is available and accessible with the help of personal computers. The purpose of the review is to inform pediatricians and public health professionals about this technique and its practical applications for designing sound nutritional recommendations. The hope is that this review will encourage the adoption of linear programming by international health professionals to improve the nutritional status of children.

1. Introduction:

* The passage discusses the importance of providing complementary nutrient-dense foods to breastfed children after 6 months of age to cover their additional energy and micronutrient requirements. Although it's assumed that suitable nutrient-dense foods are locally available to provide a nutritionally adequate complementary diet, this may not be the case. Children in this age group have high nutritional requirements and are vulnerable to multiple nutritional deficiencies, which is compounded by the need to design a nutritionally sound diet at low cost. The possibility of providing an affordable balanced diet during the complementary feeding period has never been rigorously evaluated, but a

mathematical approach called linear programming can address whether it's possible to design a diet suitable for the complementary feeding period using locally available foods and what is the minimum budget needed to cover the nutritional requirements of at least 97% of children.

* The passage discusses the history and application of linear programming in various fields of applied research and its potential use in human nutrition. The complexity of mathematically modeling food selection practices and the lack of accessible computer technology had previously hindered the practical use of linear programming in this field.

DIET DESIGN WITH AND WITHOUT LINEAR PROGRAMMING ASSISTANCE

The passage compares the traditional "trial and error" approach to designing complementary feeding diets with the linear programming approach. The traditional approach involves iterative attempts at different food combinations based on informed guesses, which can be time-consuming and may not result in an optimal solution. In contrast, once the necessary background information is collected, linear programming quickly and efficiently provides an optimal solution. It also indicates the types of foods needed to balance the diet and the cost of designing a nutritionally sound diet. However, designing a diet during the complementary feeding period is still challenging due to uncertainties regarding nutritional requirements and the absorption of different micronutrients. Linear programming offers a more reliable and efficient approach compared to the traditional "trial and error" method, which can introduce additional uncertainty and errors by trying to solve a complex mathematical problem by hand.

WHAT IS LINEAR PROGRAMMING? SOME DEFINITIONS

The passage describes the mathematical principles underlying linear programming. Linear programming is a tool used to optimize a linear function of a set of decision variables while respecting multiple linear constraints. The objective function Y to be optimized can be graphically represented by a set of parallel lines or plane surface areas, and the variables X1, X2, ... Xn whose value can be changed to optimize the function Y are called decision variables. A function Y is considered linear when it can be expressed as

$a0 + a1.X1 + a2.X2 + ... + an.Xn,$

where a0, a1, a2 ... an are constants. Similarly, a constraint on several variables is linear when it can be expressed as

$b1.X1 + b2.X2 + ... + bn.Xn \leq b0,$

where b0, b1, b2 ... bn are constants. The appendix provides a simple graphic illustration of the theory underlying linear programming to provide a conceptual understanding of the mathematical principles.

DEVELOPING A LINEAR PROGRAMMING MODEL FOR DIET OPTIMIZATION

The question that has to be answered must first be properly defined in order to create a linear programming model for diet optimisation. The objective function that provides the most useful insight into this issue must then be developed and stated as a linear function of the choice variables. Lastly, the

It is necessary to identify the dietary and palatability restrictions that must guide the diet optimisation procedure. The latter are detailed below and are essential to guaranteeing a realistic outcome.

Objective Function and Decision Variables

The passage describes how the objective function is used in linear programming to select the optimal solution from among all alternative solutions. The objective function chosen depends on the question posed. For the two key questions related to designing complementary feeding diets, the objective function will be the energy content of the diet for the first question and the total cost of the diet for the second question. In both cases, the objective functions will be minimized and can be expressed as linear functions of individual food weights, which are the decision variables. The term "food data base" is used to describe the list of all locally available foods that can be incorporated into children's diets, along with their cost and nutritional content. The weight of all locally available foods suitable for feeding young children in a specific environment is used as decision variables in the linear programming model. The term "food basket" represents the combination of foods selected during the analysis to include in the optimized diet. It represents the average amounts of different foods that should be consumed over a short period of time, not necessarily the exact amount consumed every day.

Interpreting the Results

linear programming can be used to identify potentially low nutrients in a child's diet during the complementary feeding period. The optimized diet selected by linear programming will contain these nutrients at their minimum required levels, and when the constraint on these nutrients is removed, the objective function will decrease. Iron is often a limiting nutrient for 6- to 23-month-old children, and the limiting nutrients will vary by region and season. If the linear programming analysis shows that designing a balanced diet is not possible or that the obtained diet has an unacceptably high cost or energy level, new foods with a high content of limiting nutrients should be introduced in the food database. If no suitable food can be identified, low-cost fortified food or food supplement can be used to lower the objective function and make diet optimization possible at a lower cost.

AN EXAMPLE OF DIET OPTIMIZATION BY LINEAR PROGRAMMING

how to formulate diets for 9- to 11-month-old Indian infants based on the latest WHO recommendations for complementary feeding. The objective function for minimization will be the total energy content of the diet, and the known nutritional requirements and maximum portion sizes will be the constraints. The feasibility of a solution will only exist if the minimum energy content of the food basket (including breast milk) is less than the average energy requirements of children of this age. Nutritional requirement constraints for protein, vitamins, and minerals were derived from WHO recommendations, and the calcium requirement corresponds to the latest U.S. recommendations. Breast milk was introduced in the food composition table, and an equality constraint was used in the model to guarantee its inclusion in the food basket.

USING THE COMPUTER FOR SOLVING COMPLEX DIETARY PROBLEMS:

Although direct programming may be a valuable instrument for remote wellbeing experts, selection of the method may be hampered by a need of information with the technique. The elemental activities got to create.

In this portion, a direct programming show in Microsoft Excel® is given. The as it were reason it can be found on most domestic computers and highlights a solver work that can be utilized for straight programming-based count calories optimization. This program is a viable investigation device since it empowers the

joining of various confinements and various objective capacities. Although less flexible, user-friendly modified that naturally organize information, like Nutriture, are likely ideal for hands on work purposes. To illustrate the application of straight, illustrate. and to illustrate.

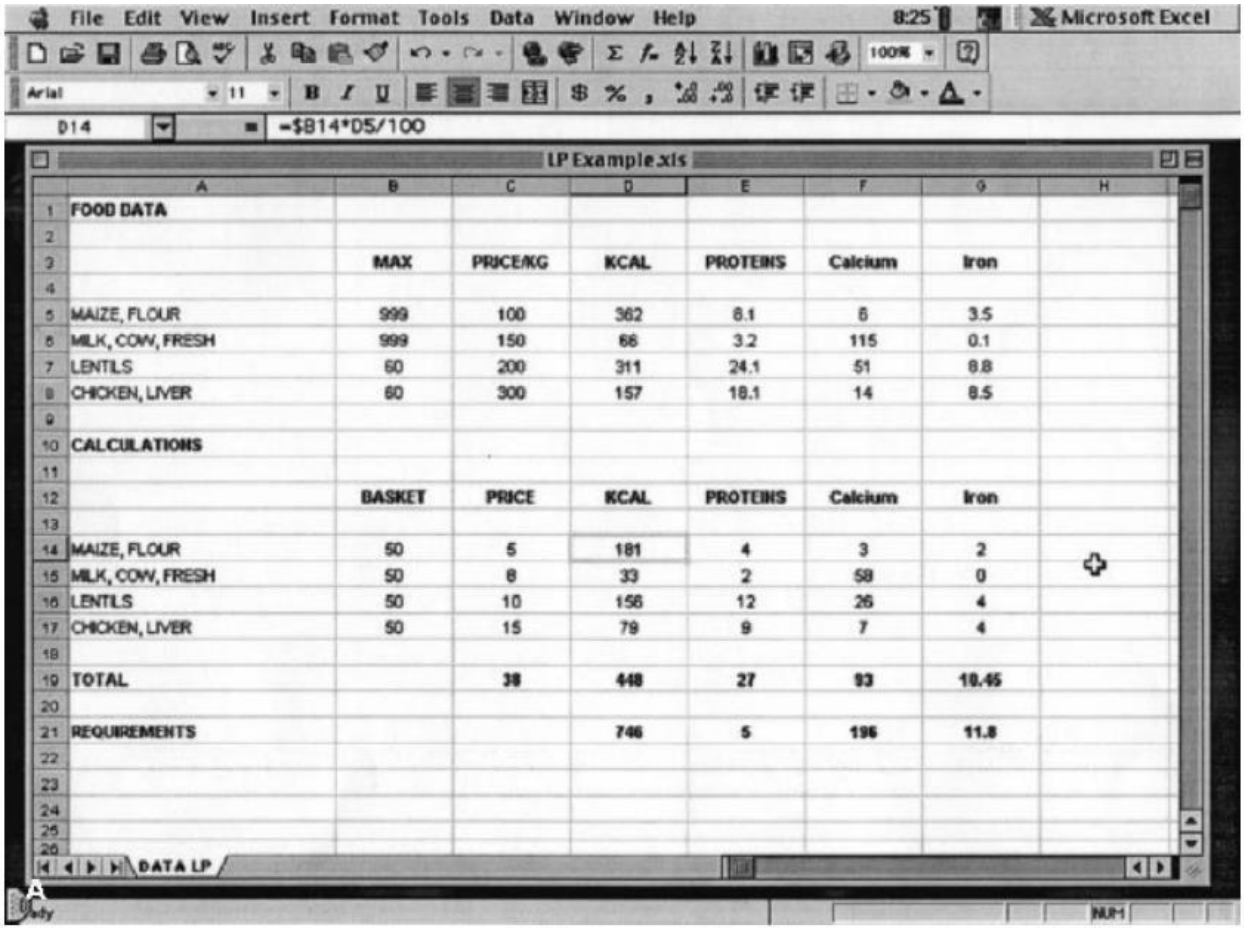
Data Layout:

In preparation for linear programming analysis, the first stage is to create a food database, as a table, The amount of nutrients, price per kilograms, and daily maximum serving size (in grammes) are listed for each food item. The daily maximum serving sizes We decided on a maximum serving size of 60 g for both lentils and liver.

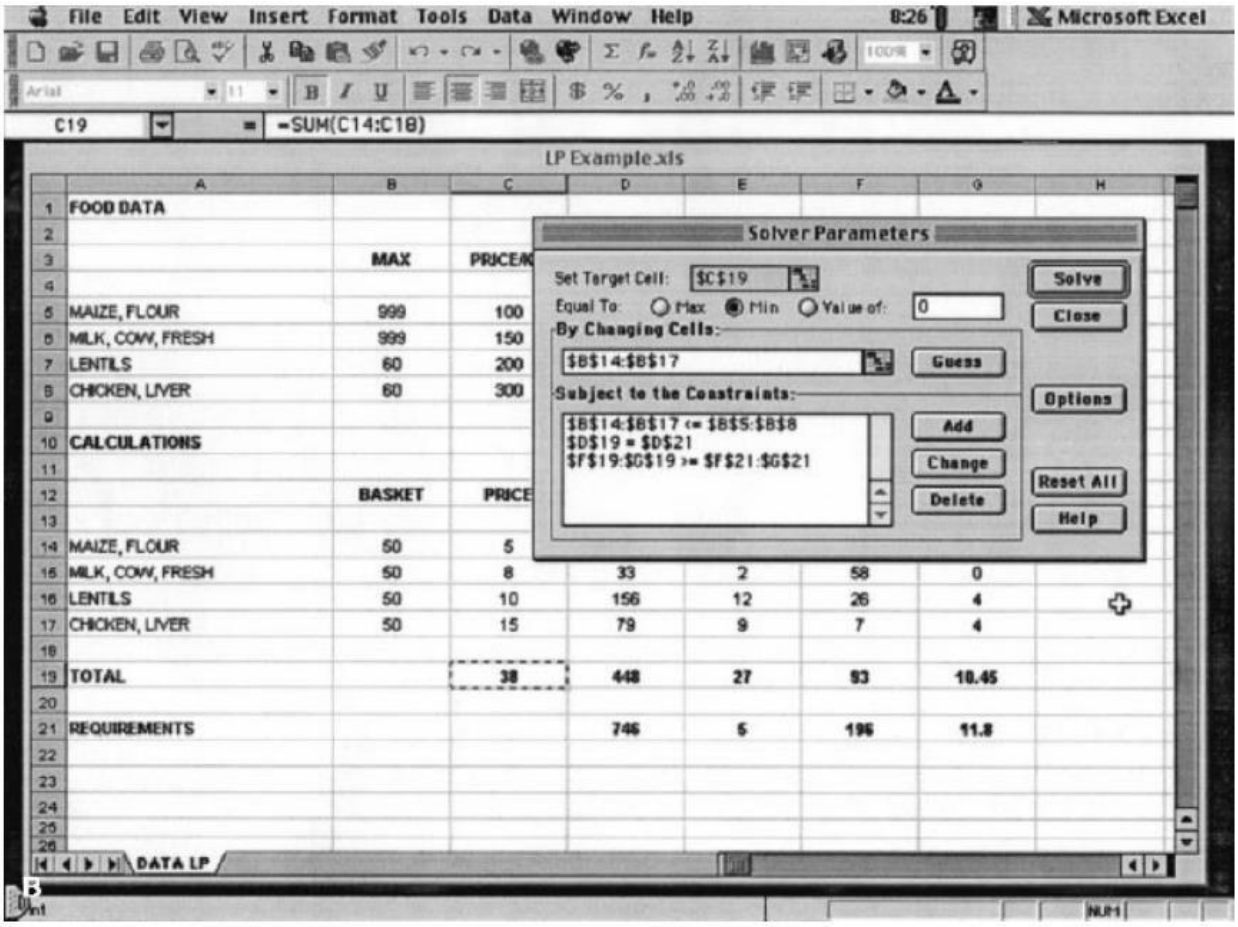
Due to their importance in the children's diets, a high arbitrary number of 999 g was chosen for maize flour and milk. The food basket with its price and nutrient content was then represented by a second table that was added immediately below the first. Based on the weights of each food item chosen, the spreadsheet determines the price and nutritional information for each food in this second table. At this point, the food basket was filled with random weights of 50 g for each food.

Activating the Solver Function and Choosing the Objective Function and the Food Variables:

As Example like figure shown blow:

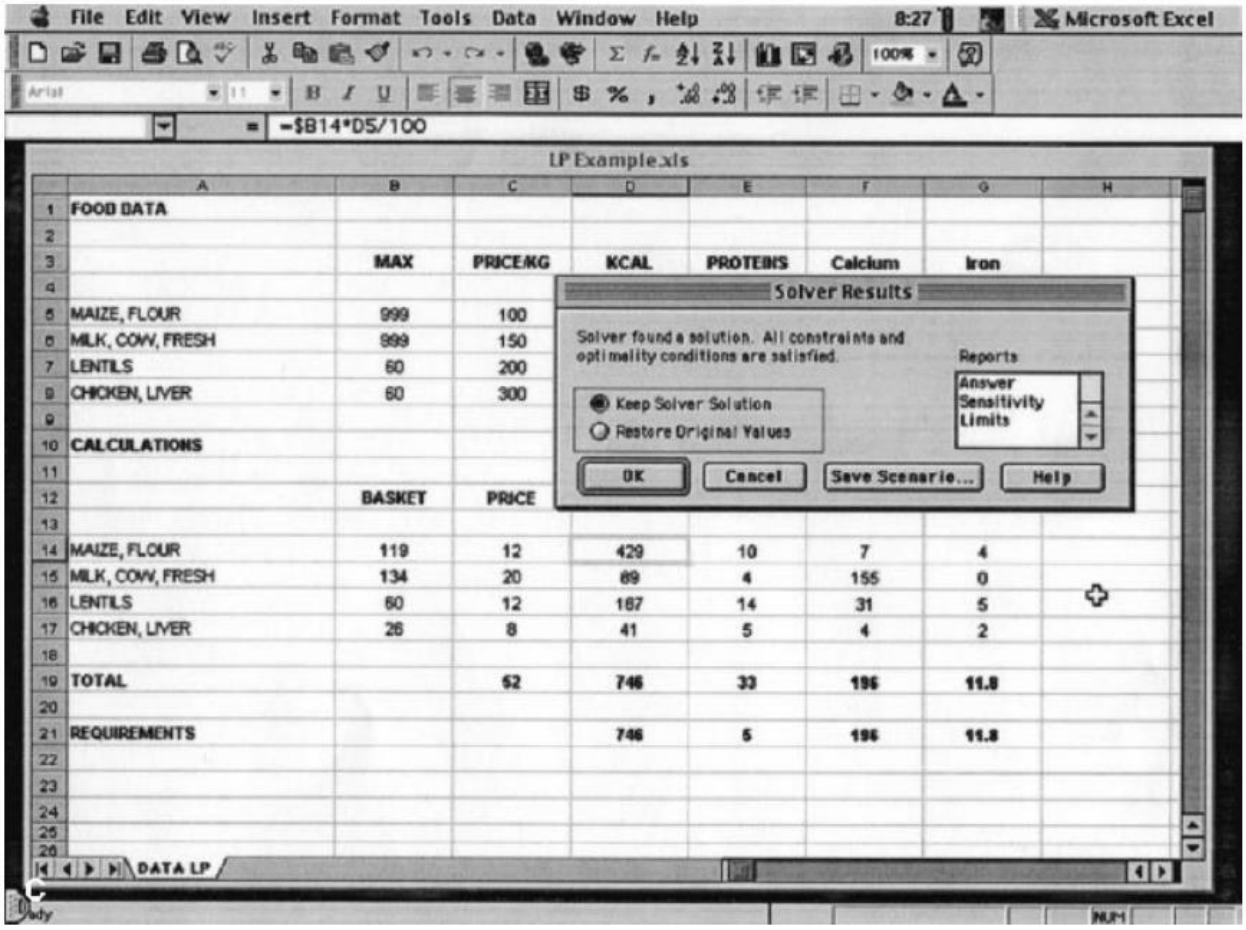


	A	B	C	D	E	F	G	H
1	**FOOD DATA**							
2								
3		**MAX**	**PRICE/KG**	**KCAL**	**PROTEINS**	**Calcium**	**Iron**	
4								
5	MAIZE, FLOUR	999	100	362	8.1	6	3.5	
6	MILK, COW, FRESH	999	150	66	3.2	115	0.1	
7	LENTILS	60	200	311	24.1	51	8.8	
8	CHICKEN, LIVER	60	300	157	18.1	14	8.5	
9								
10	**CALCULATIONS**							
11								
12		**BASKET**	**PRICE**	**KCAL**	**PROTEINS**	**Calcium**	**Iron**	
13								
14	MAIZE, FLOUR	50	5	181	4	3	2	
15	MILK, COW, FRESH	50	8	33	2	58	0	
16	LENTILS	50	10	156	12	26	4	
17	CHICKEN, LIVER	50	15	79	9	7	4	
18								
19	**TOTAL**		**38**	**448**	**27**	**93**	**10.45**	
20								
21	**REQUIREMENTS**			**746**	**5**	**196**	**11.8**	
22								
23								
24								
25								
26								



	A	B	C	D	E	F	G	H
1	FOOD DATA							
2								
3		MAX	PRICE/K					
4								
5	MAIZE, FLOUR	999	100					
6	MILK, COW, FRESH	999	150					
7	LENTILS	60	200					
8	CHICKEN, LIVER	60	300					
9								
10	CALCULATIONS							
11								
12		BASKET	PRICE					
13								
14	MAIZE, FLOUR	50	5					
15	MILK, COW, FRESH	50	8	33	2	58	0	
16	LENTILS	50	10	156	12	26	4	
17	CHICKEN, LIVER	50	15	79	9	7	4	
18								
19	TOTAL		38	448	27	93	18.45	
20								
21	REQUIREMENTS			746	5	196	11.8	
22								
23								
24								
25								
26								

A target cell must be selected first. This cell represents the price minimization goal function for optimization. As a result, is the one that needs to be optimized. Next, variable cells need to be established. These cells hold the decision factors that the program will adjust to choose the diet that satisfies all standards (for nutrition and maximum portion size) while being the least expensive. These variables in our example will be the cells that contain the weights of the maize flour, milk, lentils, and chicken liver that are ultimately included in the food basket.



Setting the Constraints and Solving the Problem:

By selecting "Add" in the constraint pane, constraints can be added. When this option is selected, a window labelled "Add constraint" appears where the cells that will be subject to the constraint are specified. Our initial restriction was to keep the total number of each food item in the food basket to no more than the maximum permitted. To make sure the total energy level of the chosen foods matched the predicted average requirements, a second restriction was applied.

By selecting the "Add" option repeatedly, additional constraints can be added to the window that says "subject to the constraints" until the total amount of protein, calcium, and iron in the diet is greater than the requirements.

Tonally into sound food-based guidelines based on locally accessible foods and local market pricing. totally accepted nutritional recommendations. In fact, linear programming is far more effective at creating diets during the supplemental

feeding period than the present empirical "trial and error" method. As a result, it might become a crucial tool for pediatricians and those in charge of developing nutrition programmed.

A GRAPHIC INTRODUCTION TO LINEAR PROGRAMMING:

The approach utilized by direct programming to reply.

the questions raised within the presentation with respect to feasibility and taken a toll of eat less detailing can be outlined by a

straightforward illustration based on a slim down made of two nourishments. This basic, nonrealistic, hypothetical illustration was chosen because it can be represented in a two-dimensional chart.

To begin with we are going look at graphically how direct programming decides whether calcium and press requirements can be met when a complementary slim down is based on dairy animals drain and maize flour.

The calcium prerequisite can be met, while press requirements cannot be met by giving maize and dairy animals drain.

This example, therefore, outlines the two conceivable outcomes of a direct programming investigation, specifically that. planning to count calories is or isn't doable.

lowed by a graphic illustration of the method used by linear programming to identify the combination of two foods meeting two nutritional constraints at the lowest possible cost.

CONCLUSIONS:

Linear programming speaks as a capable instrument for planning optimized diets for youthful children amid the complementary bolstering period.

It may be a valuable scientific strategy that can be utilized to effortlessly interpret universally acknowledged wholesome proposals into sound food-based rules based on locally accessible nourishments and neighborhood showcase costs. Undoubtedly, straight programming is much more proficient than the

observational “trial and error” approach right now utilized for defining diets amid the complementary nourishing period. As such, it seems to be an irreplaceable apparatus for pediatricians and sustenance program organizers.

REFERENCES:

1. Brown KH, Dewey KG, Allen LH. Complementary feeding of young children in developing countries: a review of current scientific knowledge. World Health Organization. 1998. Geneva. Available at: http://www.who.int/child-adolescenthealth/New_ Publications/NUTRITION/WHO_NUT_98.1.pdf. Accessed September 10, 2002. 2. World Health Organisation. The optimal duration of exclusive breastfeeding. Report of an expert consultation. 28-30 March 2001. WHO: Geneva. Available at: http://www.who.int/childadolescent health/New_Publications/NUTRITION/WHO_CAH_01_24.pdf.

Accessed September 10, 2002