Backtracking Algorithm to Arrange Food to Fit Desired Calories

Backtracking Algorithm Application

Dwiani Yulia Ariyanti/13518142
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13518142@std.stei.itb.ac.id

***Abstract*—This paper described the program prototype to find the highest value of combining ingridients of various food using backtracking algorithm. Problem solving in this problem is like 1/0 Integer Knapsack, and solved by the same way as 0/1 knapsack problem. This program made to help people that want to organize their calories consumption but still get the best value of protein.**

Keywords—calories, backtracking, food, health

I. Introduction

Calories are a measure of energy(cal), and more often used in kilogram unit, where 1 kcal equals 1000 cal . In daily life, calories uses to maintain people's health and adjust their activity's calories requirements. In special case, people try to arrange their calorie trough food intake to maintain body weight for perfect body proportions or just in way to recover from diase.

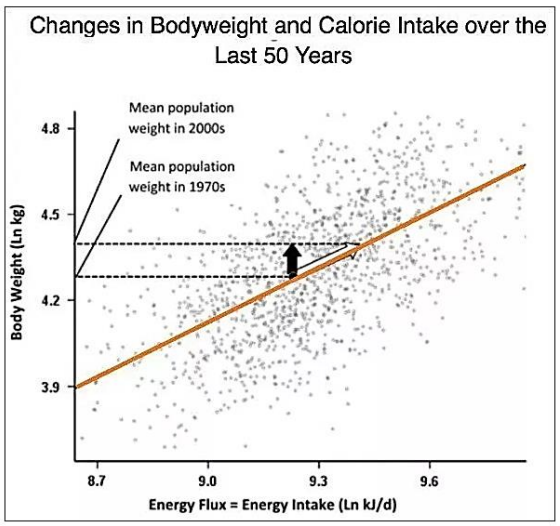

Figure 1.Changes in Bodyweight and Calorie Intake Represented in Graph (https://www.healthline.com/nutrition/7-graphs-prove-calories-count#section1)

Food is needed by our body to maintain energy to do any activities. Calories that contain in food, suply our body with the energy needed to sustain life. "Calories" written by Eva V. Osilla and Sandeep Sharma said that each cell in our body need energy to carry out their specific tasks, from protein metabolism to the Krebs cycle. When we eat foods, they are broken down to release this energy which is either used by the body immediately or stored for later use, depending on the body's needs at the time. Generally, we need food to live our life.

The problem is, as time goes by, people start doing diet system to maintain their body weight and body proportion. But this habbit are often unhealthy and actually endanger their bodies itself. In this paper, we will make a simple application that will provide a list of food from user's input that contain variations of food that can be combined to make a certain number of calories. So they who does diet system will just know how many calories in certain that they eat, not just eat less for keep their body proportion. In addition, it will also provide variant combination of food, so that they will not getting bored with these low calories foods. And not for only them who do diet, it help ordinary people to fulfill their calories in certain too with some combination according to the ingredients they just have.

II. Theoritical Framework

A. Optimization Problem

In optimizing design, its objective need to be simply so we can minimize the cost of production or maximize production's efficiency. An optimization algorithm is a procedure, executed iteratively by comparing various solutions, untill we found the optimum solution. Optimization has become a part of computer-aided design activities since computers came.

There are two distinct types of optimization algorithms widely used today, deterministic algorithm and stochastic algorithm. In deterministic algorithm, we use specific rules for moving one solution to other. These algorithms are in use to suite some times and have been successfullyapplied for many engineering design problems. While stochastic algorithms are in nature with probabilistic translation rules. These are gaining popularity due to certain properties which deterministic algorithms do not have.

According to references [4], we know that in naive optimazing design, we use a priority problem knowledge to comparing a few limited alternative solutions that created by it. Feasibility of each design solution is first investigated. An estimate of underlying objective such as cost and profit of each solution is compared, then we adopted the best solution. It is impossible to apply single formulation procedure for all engineering design problems, since the objective in a design problem and associated therefore, design parameters vary product to product different techniques are used in different problems. Purpose of formulation is to create a mathematical model of the optimal design problem, which then can be solved using an optimization algorithm.

B. 0/1 Knapsack Problem

The name "knapsack problem" dates back to the early works of mathematician Tobias Dantzig (1884–1956), refers to the commonplace problem of packing the most valuable or useful items without overloading the luggage. Knapsack problem is kind of combinatorial optimazion. There will be set of items that given, each item has it own weight and value. We need to find lergest value from these items collection that less thangiven limit weight. Integer knapsack problem can be represented by figure 2.

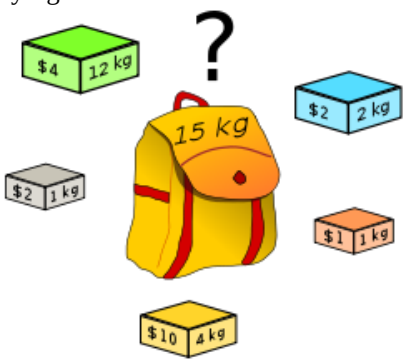

Figure 2. Integer Knapsack Problem Representation

(https://upload.wikimedia.org/wikipedia/commons/thumb/f/fd/Knapsack.svg/486px-Knapsack.svg.png)

For example, in figure 2, the bag only can added by item that have total weight 15 kg. We need to find item combination that have the most value as result. It also possible to have more than one combination. We can use the equation below to representing the solution:

maximizes $\sum_{i=1}^{n} v_i x_i$

subject to $\sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}$

where:

v_i represent tuple of item's value $< v_1, v_2,...,v_n >$

w_i represent tuple of item's weight $< w_1, w_2,...,w_n >$

x_i represent tuple from copies of each kind of item to zero or one $< x_1, x_2,...,x_n >$

For bounded knapsack problem, we removes the restriction that there is only one of each item, but restricts the number of x_i copies of each kind of item to a maximum non-negative integer value c:

maximizes $\sum_{i=1}^{n} v_i x_i$

subject to $\sum_{i=1}^{n} w_i x_i \leq W \text{ and } 0 \leq x_i \leq c$

While in unbounded knapsack problem, we places no upper bound on the number of copies of each kind of item, can be formulated as above, except for restriction on x_i is that it is a non-negative integer. So the equation become:

maximizes $\sum_{i=1}^{n} v_i x_i$

subject to $\sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \geq 0$

Integer Knapsack Problem can be solved by brute force algorithm by try all 2^n possible subsets n. But it can be optimized using other algorithm such as divide and conquer algorithm, dynamic programming, backtracking algorithm, etc. In this paper, we will use backtracking algorithm to optimizing the solution.

C. Backtracking Algorithm

Backtracking algorithm introduced in 1950s by D.H. Lehmer. Then R.J Walker, Golomb, and Baumert represents a general description of this algorithm. Backtracking is kind of algorithm where just lead to solutions that explored, we do not considered about choices that don't lead to solutions. If we represent this by a graph, nodes that contain unconsidered choices will be prunned. It's the improvement of exhaustive search by brute force algorithm that need to find all the possibilities of subset and compare it one by one to get the best choices.

Based on references[2], general properties of backtracking methods can be described as follows:

- Set of solution

  The solutions are represented by vectors with m-tuple: $X = (x_1, x_2, ..., x_m)$, where $x_i \in S_i$.

  S_1 can be equals to S_2 as well as S_n. For example we have S_i:

  Si = {0, 1}, so x_i is whether 0 or 1.

- Generator Function for x_k's value / T(k)

  T (k) generates a value for x_k which is a vector component of the solution.

- Border function / B(x_1, x_2, ..., x_k)

  We assign B with true values if (x_1, x_2, ..., x_k) leads to the solution. If we get true, then continue the generation of values for x_{k+1}. But if false, then (x_1, x_2, ..., x_k) discarded.

All possible solutions to the problem are called solution space and can be represented as tree structure, which each tree node represented states of the the problem, and the branch represented as x_i's values. The trajectory that lines from root to leaf of the tree represents the possible solution, and the entire trajectory from root to leaf forms solution space. Organizing the solution space tree referred to as state space tree.

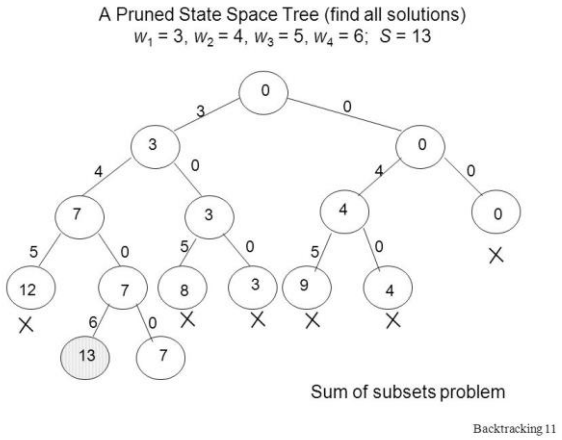

Figure 3. Representation of Backtracking Algorithm by Tree

(http://pianetamedia.com/ppt-knapsack-problem-solving-using-backtracking-using-state-space-tree-76/)

According to references[2], the principle of finding a solution with backtracking method can be written is as follows:

- Solutions are searched by forming a trajectory from root to leaf. The rules of formation are used is following the depht-first order (DFS) rule.
- Nodes that have been generated are called live-node.
- Live-nodes that are being expanded are called node-E (Expand-node).
- Each time the E-node is expanded, the trajectory is built by increasing in length.
- If the path being formed does not lead to the solution, then the E-node is "killed" / "pruned" so that it becomes a dead node.
- Border function used to kill the E-node that not quantify the function.
- Dead node will never be expanded again.
- If the formation of the path ends with a dead node, then process other searching for backtrack to the node above it
- Then, continue by generating the other child node, this node becomes the new E-node.
- It stopped when we have reached goal node.

Backtracking solved three types of problems:

1. Decision Problem, which aim to search for a feasible solution.
2. Optimization Problem, to search for the best solution.
3. Enumeration Problem, in purpose to find all feasible solutions.

D. Food Composition

Written in "Food composition data" book by H. Greenfield and D.A.T. Southgate, early food composition studies were carried out to identify and determine the chemical nature of the principles in foods that affect human health. But studies are still required, because current knowledge of nutrition is still incomplete, and often at an ever increasing level of sophistication, into the composition of foods and the role of these components and their interactions in health and disease. Somogyi (1974) reproduced a page of the earliest known food composition table, dated 1818. Ever since, it has been customary to record food composition data in printed tables for use by both specialists and non-specialists. While printed tables will continue to be produced, computerized data systems have replaced them in some settings because of the ease with which data can be stored, and the facility with which the large amounts of data can be accessed and processed. These table will be our source data, and it will be better if we organize it into basis data, so we can use it to compute a program that need food composition to be their data.

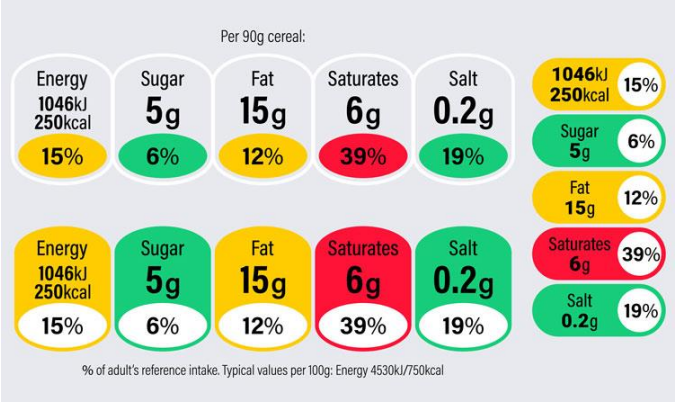

Figure 4. Lebalelling Food, to Measure The Sustention

(https://www.newfoodmagazine.com/news/78337/food-labels/)

Every food contain various value of kalories, protein, vitamin, water, and minerals. It used to fulfill our body's need so we can do our activity and keep our body healthy and fit. Too much calories can bring in disease, like obesity. Too less calory can also cause hungry edema and make our body not fit enaugh to do activity properly. So what we need is the balance kalories, protein, water, mineral, and various vitamin consumed by our body.

For athlete, model, or people that need to control their calorie consumption (it can be for treatment during illness too), they still need carbohydrate source to gain energy for their daily activities. But in other hands, they need to minimilize

their kalories consumption. We can manipulated it using protein consumption, because protein is actually also carbohydrate source, beside it's give our body more healthy and most of high protein food contain low fat. It more healthy, and help us to reduce calories consumption.

There are many food that contain low calories beside high protein. The problem that we want to solve is, how to gain enaugh energy but consume less calories. What kind of food, and how to combinate it to get the highest protein contained, while under calories limit we want.

III. Implementation

In this paper we will modelling the problem using 0/1 knapsack problem. The equation will be in follows:

maximizes $\sum_{i=1}^{n} v_i x_i$

subject to $\sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}$

where:

v_i represent tuple of food's protein value $< v_1, v_2,...,v_n >$

w_i represent tuple of food's calorie value $< w_1, w_2,...,w_n >$

x_i represent tuple from copies of each kind of item to zero or one $< x_1, x_2,...,x_n >$

The data that used in this program is from txt. It will be better if we can use database like MySQL, but for this simple prototype, we eill just use txt file. So, we need to extract this data into array. We will use command as follows:

```
def bacaFile():
    file = open("calories_info.txt","r")
    matriks = []

    for line in file:
        row = [elt for elt in line.split(' ')
]
        matriks.append(row)

    for i in range(len(matriks)):
        matriks[i].append(False)

    file.close()

    return matriks
```

We saved and process the data that represented in matrix. Matrix in column 1 saved the food type, column 2 saved the calory value, and column 3 saved protein value.

Because backtracking implements dfs, we will modification the algorithm for dfs as bellows:

```
def dfs(visited, graph, node):
    if node not in visited:
        print(node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)
```

The dfs algorith will be modificated into backtracking algorithm. The root will be an empty list. Then we will generate the branch one by one with make 2 node where has addition value True/False. It represented 0 and 1 value for x_i.

```
def backtracking(limit,node, curr_food, next_
food, curr_weight, food_info):

    # node active = node saat itu, semua dimu
lai dari true
    node.pop()
    if(getFoodInfo(curr_food,food_info)[5]):
        if (curr_weight + getCalorie(curr_foo
d,food_info) > limit):
            curr_weight += getCalorie(curr_fo
od,food_info)
            take = getFoodInfo(next_food,food
_info)
            print(take)
            take[5] = True
            throw = getFoodInfo(next_food,foo
d_info)
            throw[5] = False
            node.insert(0,throw)
            node.insert(0,take)
    else:
        take = getFoodInfo(next_food,food_inf
o)
        print(take)
        take[5] = True
        throw = getFoodInfo(next_food,food_in
fo)
        throw[5] = False
        node.insert(0,throw)
        node.insert(0,take)
    return node, curr_weight
```

We will just pruned the nodes if it has over the limit. Then we will go to its parrent and check the other child of it if it hasn't checked yet. The backtracking will run untill every live-nodes and expand-node empty. Then we will get the highest protein value from set of solution.

```
inp = input()
food_desired.append(inp)
while (inp != "."):
    inp = input()
    if (input != "."):
        food_desired.append(inp)

curr_weight = 0
for i in range(len(food_desired)-1):
    (node, curr_weight) = backtracking(desire
d_calories,node,food_desired[i],food_desired[
i+1],curr_weight,food_info)

(node, curr_weight) = backtracking(desired_ca
lories,node,food_desired[len(food_desired)-
1],food_desired[len(food_desired)],curr_weigh
t,food_info)
```

For the input user will input the food ingridients that also wil be showed before. And end it with write dot (.). Then user will input the limit of calories that they want. We will compute every desired food that written by user one by one using backtracking function and wait for the result.

IV. Case Studies

For case studies we use data as follows:

Food	Calorie	Protein	Weight(gram)
Wortel	41	0.93	100
Tomat	18	0.88	100
Nasi	129	2.66	100
Roti	266	7.64	100
Putih telur	52	10.9	100
Alpukat	160	2	100
Kentang	70	1.68	100
Selada	14	0.9	100
Timun	12	0.59	100
Almond	578	21.26	100
Susu	50	3.29	100

1) We will try to insert some ingridients for example: wortel, nasi, tomat, almond, susu. Then let 50 being the calorie limit. Then we will search the highest protein contained in food combination. For this cas, we got susu as the result.

```
dwianiyulia@dwianiyulia-X442URR:/media/dwianiyulia/Mee/Life/Stima/Makal
ah$ python3 calories_aranger.py
food weight(gram)
wortel 100(gram)
tomat 100(gram)
nasi 500(gram)
roti 100(gram)
putihtelur 100(gram)
alpukat 100(gram)
kentang 100(gram)
selada 100(gram)
timun 100(gram)
almond 100(gram)
susu 100(gram)
Masukkan daftar makanan yang Anda miliki: (ketikkan tanda titik jika su
dah selesai)
wortel
nasi
tomat
almond
susu
.
Masukkan besar kalori yang diinginkan: 50
susu 100 gram
Total calorie: 50
Total protein: 3.29
```

Figure 5. Case Studie 1

2) We will try to insert some other ingridients: nasi, susu, kentang, selada, alpukat, almond, tomat, and wortel. Then let 100 being the calorie limit. Then we will search the highest protein contained in food combination. For this cas, we got susu, selada, and tomat as the result.

```
dwianiyulia@dwianiyulia-X442URR:/media/dwianiyulia/Mee/Life/Stima/Makal
ah$ python3 calories_aranger.py
food weight(gram)
wortel 100(gram)
tomat 100(gram)
nasi 500(gram)
roti 100(gram)
putihtelur 100(gram)
alpukat 100(gram)
kentang 100(gram)
selada 100(gram)
timun 100(gram)
almond 100(gram)
susu 100(gram)
Masukkan daftar makanan yang Anda miliki: (ketikkan tanda titik jika su
dah selesai)
nasi
susu
kentang
selada
alpukat
almond
tomat
wortel
.
Masukkan besar kalori yang diinginkan: 100
susu 100 gram
selada 100 gram
tomat 100 gram
Total calorie:  82
Total protein:  5.07
```

Figure 6. Case Studie 2

V. Conclusion

0/1 Knapsack Integer Problem can be applicated in many domain. For this paper we use it to find the highest protein that can be contained in food combination with the calorie limit that we want. Backtracking algorithm will pruned the node that not lead into solution, so it make this program faster than compute it using exhaustive search.

Video Link at Youtube

https://youtu.be/MdpULnfTXZk

Acknowledgment

The author would like to thank God for all the grace and blessings. The author would also thank Dr. Masayu Leylia Khodra, Dr. Ir. Rinaldi, M.T., and Dr. Nur Ulfa Maulidevi as the lecturer of algorithmic strategy (IF2211) for all the knowledge given. The author also express gratitude to family and friends for their support in process of making this paper.

REFERENCES

[1] Eva V. Osilla, Sandeep Sharma, "Calories", 2018, StatPearls Publishing LLC.

[2] R. Munir, "Algoritma-Runut-balik (Backtracking)", 2020, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Algoritma-Runut-balik-(2020).pdf

[3] Lecture 13: "The knapsack problem", http://www.es.ele.tue.nl/education/5MC10/Solutions/knapsack.pdf

[4] "Optimization methods", https://mech.iitm.ac.in/nspch52.pdf

[5] H. Greenfield and D.A.T. Southgate, 2003, "Food composition data", 2nd ed, Food and Agriculture Organization of the United Nations Rome

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2020

Dwiani Yulia Ariyanti - 13518142