Game Strategy: Optimizing Strategy Using the Binary Knapsack Algorithm
Tommy Yang
Menlo School

Game Strategy: Optimizing Strategy Using the Binary Knapsack Algorithm

1. Abstract

Score maximization is critical to a team's performance during competition and is thus one of the most important elements of overall Botball game strategy. The design of an optimal task list that maximizes combined task score while still remaining within the time limit is central to Botball scoring strategy. This paper presents an overview of the knapsack problem, its common variations, and real-world applications. In addition, it presents a research model that uses the binary knapsack algorithm to produce an optimal task list to score the most points based on the 2017 regional Botball scoring rubric. The paper concludes that the use of the knapsack algorithm to decide tasks performed is highly beneficial in improving a robot's maximum theoretical score.

2. Introduction

In Botball, a team's seeding rank is based on the team's two highest seeding scores. Therefore, teams must take time to optimize the tasks their robots complete in order to maximize their point potential.

However, the reality is that most Botball teams, including ours, lack a systematic approach to their scoring strategies. Teams select their task lists through estimation using past experience, gut feel, and trial-and-error. These methods have lead to teams operating in the blind, not knowing if their implementations are optimal.

The goal of the study is to effectively select a subset of game tasks with a fixed time constraint (2 minutes) so that the sum of the scores of selected tasks is maximized.

The Botball score maximization problem falls in the area of combinatorial optimization, which aims to find the optimal combination of objects from a finite set of objects. There are many mathematical algorithms that can be leveraged for score maximization within the realm of combinatorial optimization. The algorithmic solution to the knapsack problem is one of the most classic and powerful algorithms that can solve this problem; thus, it is the focus of this paper.[1]

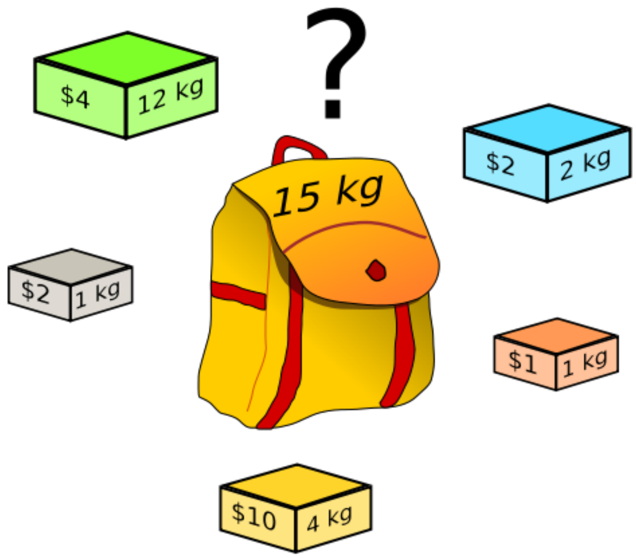
2.1 What is the Knapsack Problem?

As a well-known problem in combinatorial optimization, the knapsack problem derives its name from the following ancient story:

> A long time ago, a thief broke into a vault chock full of valuable items. However, he only brought one knapsack with him and must fill it with the most valuable items without breaking it. The vault has n items, where item *i* weighs s_i pounds, and can be sold for v_i

[1] Skiena, S. September 1999. Who is Interested in Algorithms and Why? Lessons from the Stony Brook Algorithm Repository. AGM SIGACT News. 30 (3): 65–74. doi:10.1145/333623.333627.

dollars. The thief must choose which items to take in his knapsack so that he makes as much money as possible after selling the items.



Example of a one-dimensional (constraint) knapsack problem: Which boxes should be chosen to maximize the amount of money while still keeping the overall weight under or equal to 15 kg? (Illustration of the knapsack problem by Duke / CC BY-SA 2.5).

With works dating as early as 1897, the knapsack problem has been studied by mathematicians for over a century.[2] The original knapsack problem can be rewritten as follows:

Given a finite set of items which have a weight and a value, choose the subset of all items such that the total weight is less than or equal to the maximum weight the knapsack can hold and the total value is maximized.[3]

2.2 Knapsack Problem Variations and the Binary Knapsack Algorithm

There are a few variations of the knapsack problems, such as multi-objective knapsack problems, multidimensional knapsack problems, multiple knapsack problems, quadratic knapsack problems, and subset-sum problems. In the real-world, the most common problem is the 0-1 knapsack problem, also known as the binary knapsack problem, because in most practical use cases, a binary “yes” or “no” decision must be made regarding a task. In the story of the thief in the vault, the thief must decide to either take an item or leave it behind as the items are assumed to be indivisible. Thus, the story is an example of a binary knapsack problem.

[2] Mathews G. B. June 1897. On the partition of numbers. Proceedings of the London Mathematical Society. 28: 486–490. doi:10.1112/plms/s1-28.1.486.

[3] Cormen T, Leiserson C, Rivest R, Stein C. 2009. Introduction to Algorithms. 3rd edition. Cambridge (MA): MIT Press. Section 16.2: The 0-1 Knapsack problem, p.425-428.

The National Institute of Standards and Technology (NIST) formally defines the binary knapsack problem as follows.

> "There is a knapsack of capacity $c > 0$ and N items. Each item has value $v_i > 0$ and weight $w_i > 0$. Find the selection of items ($\delta_i = 1$ if selected, 0 if not) that fit, $\sum_{i=1}^{N} \delta_i w_i \leq c$, and the total value, $\sum_{i=1}^{N} \delta_i v_i$, is maximized."

The simplest solution to the binary knapsack problem is to simply try out every case. Every object can be either included or not included, which leads to a $O(2^N)$ runtime. While we do not really care about speed and our N is relatively small, it would be interesting to look at a solution that is more time efficient, since this implementation recalculates some values multiple times and looks at impossible cases when the total time exceeds the time limit. We can use dynamic programming to create a more efficient algorithm, which results in a $O(c*N)$ runtime.

2.3 Knapsack Algorithm's Applications

According to a study by Stony Brook University, the knapsack problem ranked as the 18th most popular and the 4th most needed out of 75 algorithmic problems.[4] It is not surprising that the knapsack algorithms are widely used in a variety of real-world applications. Here are some interesting applications of the knapsack algorithms:

- In manufacturing, it is used to find the least wasteful way to cut raw materials
- In investment banking, it is used to optimally select stocks for maximum portfolio returns
- In cryptography, it is used to generate keys for Merkle–Hellman knapsack public key cryptosystems[5]
- In computing, it is used by a server to pack optimal chunks of data in one go during a software download to fully utilize the size limit
- In space travel, it is used to maximize the values of goods that a spaceship carries.

3. Binary Knapsack Algorithm's Application in Botball Scoring

The knapsack algorithm can also be used by Botball teams to formulate the optimal scoring strategy. Each task can be associated with a score and the time it takes to complete the task. Here is how to apply the binary knapsack algorithm to Botball:

- Task completion time replaces item weight
- The time limit of 2 minutes replaces knapsack weight limit
- Task score replaces item value

The goal of using the knapsack algorithm in Botball is to figure out how to achieve high scores by carefully choosing a set of tasks that fit within the game's time limit.

---

[4] Skiena, S. September 1999. Who is Interested in Algorithms and Why? Lessons from the Stony Brook Algorithm Repository. AGM SIGACT News. 30 (3): 65–74. doi:10.1145/333623.333627.

[5] Merkle R, Hellman M. 1978. Hiding information and signatures in trapdoor knapsacks. Information Theory, IEEE Transactions on. 24 (5): 525–530. doi:10.1109/TIT.1978.1055927.

3.1 Research Mythology

At the start of this project, I determined that score maximization was a critical part of game strategy and that it was best to develop the optimal task list before designing the robot. Although there were other considerations in the Botball game that contribute to scoring, it was best to ignore them during the abstraction process to simplify the research model. Thus, I developed the following research model to generalize Botball score maximization:

- A starting point
- A destination
- One robot is used to conduct the tasks
- Task completion time is estimated
- Only the highest scoring task in a category is considered to simplify the model
- Task sequence and travel time/shortest route are ignored in this study, but its effect can be further studied
- The two robot use case is ignored in this study, but its effect can be further studied
- Point values from the regional competitions will be used

Based on the above research model, I designed a project that uses the binary knapsack algorithm to accomplish the optimal set of tasks that results in the maximum possible score. In future studies, I can adjust the solution by overlaying other considerations or adding other constraints to satisfy ignored conditions.

3.2 Research Model Implementation

This Year's Game

Robots Assisting a Modern Agricultural Operation

Managing a modern agricultural operation is hard work, but with the use of robotic technologies, the operations can become more efficient in their cultivation and use of resources, in particular water and fertilizer. It is planting time and Agrobot has just finished getting the family farm ready. The *Barn* has been cleared and the seed *corn* loaded into a *Seed Bin*. There is a *Greenhouse* on the *Terrace* that Agrobot has made ready for seedlings to be planted in its nursery. Seedlings have been laid out on a bench on the floor of the greenhouse for the *Green House's Lower Planter*, and more have been laid out on the *Shelf* for planting in the *Upper Planter*. *Hoppers* near the *Fields* have been loaded with fertilizer. Water has been pumped from the *Aquifer* to fill the W *Tanks*. The *Fields* have been plowed into *Furrows* and are ready to be planted with seed corn.

To help Agrobot complete preparation of the *Fields* for planting, your robots need to remove *hay bales* so planting can commence, preferably stacking them in the *Barn*. There is a *cow* that your robots need to return to the *Grazing Area* or put in the *Barn*. Your robots will be in charge of planting the seed corn in the *Field* along with the correct amount of fertilizer and water to help produce the maximum amount of yield at harvest time. In addition, your robots need to plant the seedlings in the greenhouse planters in case there is a late freeze. The seedlings also need to have the right combination of fertilizer and water to thrive. If the long-range weather forecast is okay, seedlings can be planted in the *Field* with the corn. To help your robots adapt to the job, Agrobot has provided samples of seed corn, seedlings, fertilizer, water, and a hay bale to use when you deploy your robots.

Based on the above research model, I implemented the project through the following steps:

Step 1: Compile the complete list of all tasks associated with a score

The following list is a simplified list of all tasks associated with a score:

Task	Point value	Time (seconds)
Robot on terrace	25	20
Botguy and 2 cows on terrace	300	90
2 water containers in upper planter	120	80
Pink poms in upper planter (12 poms)	120	40
Orange poms in upper planter (12 poms)	120	60
Blue poms in upper planter (12 poms)	120	80
Green poms in upper planter (12 poms)	120	60
Sorted poms (3 furrows, 4 colors, 12 poms per furrow)	1440	200
Stacked hay in barn (7 high)	490	110

Step 2. Apply the binary knapsack algorithm using the following Python code

```
def binary_knapsack(T, time, pts, n):
    K = [[0 for x in range(T+1)] for x in range(n+1)]
    for i in range(n+1):
        for w in range(T+1):
            if i==0 or w==0:
                K[i][w] = 0
            elif time[i-1] <= w:
                K[i][w] = max(pts[i-1] + K[i-1][w-time[i-1]],  K[i-1][w])
            else:
                K[i][w] = K[i-1][w]
    return K[n][T]


pts = [25, 300, 120, 120, 120, 120, 120, 1440, 490]
time = [20, 90, 80, 40, 60, 80, 60, 200, 110]
T = 120
n = len(pts)
print(binary_knapsack(T, time, pts, n))
```

Step 3. Read the output

The code outputs 490, which means stacking 7 hay bales in the barn will give the optimal score for one robot.

3.3 When to Use the Binary Knapsack Algorithm in the Botball Season

In Botball, the determination of target robot tasks dictates critical requirements for robot design, task sequencing, and route planning. Therefore, it is best to develop the targeted optimal task list using the knapsack algorithm at the beginning of the season before the start of robot design, task sequencing, and route planning. This sequencing order ensures the requirements for the robot design and route planning are captured and analyzed upfront by the team. This way, the team can avoid major in-season design changes, reduce the number of iterations, and maximize the operational efficiency of a team while achieving the best possible score for the season.

4. Conclusion

This study introduces knapsack problems, algorithms, and applications in the context of Botball game strategy. The binary knapsack algorithm has been successfully applied to a single robot Botball scoring model to maximize task combined scores. Through this study, students learn how the binary knapsack can be applied to achieve the highest potential score. In addition, for operational efficiency, it is important to note that a team should conduct the score maximization planning phase before robot design, task sequence, and route planning phases. Furthermore, this project demonstrates that it is necessary to learn and utilize proven combinatorial optimization algorithms and use them in competition to improve game results.