Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering

SMART BUDGET PLANNER

(0/1 Knapsack Problem Solver)

BACHELOR OF TECHNOLOGY
(Computer Science and Engineering.)

SUBMITTED BY:
Sagar Jaswal(23300193)
Divyanshi Dadhwal(2330099)
Geetanjali Shree(2330104)
Muskan Kumari(2420360)

Under the Guidance of
Dr. Annu
Assistant Professor
Department of Computer Science & Engineering Chandigarh Engineering College Jhanjeri Mohali - 140307

Table of Contents

Abstract:

The **Dynamic Knapsack Problem** is a fundamental combinatorial optimization problem that seeks to maximize the total value of selected items while adhering to a given weight constraint. It is efficiently solved using **dynamic programming**, where subproblems are solved and stored to avoid redundant computations. The approach constructs a **DP table** to determine the optimal selection of items using a recursive relation based on whether to include or exclude an item. With a time complexity of O(nW), this method is widely applied in resource allocation, investment planning, and decision-making processes. Optimizations such as **space reduction techniques** further enhance its efficiency for large-scale applications.

Introduction:

The Dynamic Knapsack Problem is a well-known combinatorial optimization problem that arises in various real-world applications, such as resource allocation, budgeting, and logistics. It belongs to the class of NP-complete problems and is an extension of the 0/1 Knapsack Problem, where each item can either be included or excluded from the selection. The objective is to maximize the total value of selected items while ensuring that their combined weight does not exceed a given capacity.

To solve this problem efficiently, dynamic programming is used, which breaks it down into smaller overlapping subproblems. By storing the results of these subproblems in a table and building solutions incrementally, the approach ensures an optimal solution without redundant calculations. This method significantly reduces computational complexity compared to brute-force approaches and provides an efficient way to handle large instances of the problem. The Dynamic Knapsack Problem has practical applications in economics, finance, logistics, and computer science, making it a crucial area of study in optimization and algorithm design.

Brief Literature Survey:

The **Knapsack Problem (KP)** has been extensively studied in operations research, combinatorial optimization, and computer science. It serves as a foundation for numerous real-world applications, leading to various solution approaches and optimizations.

Early Studies and Classical Approaches

- The **Knapsack Problem** was first introduced as a combinatorial optimization problem in the early 20th century. Early approaches included **brute-force enumeration**, which examines all possible combinations but is computationally infeasible for large instances.
- **Bellman (1957)** introduced the **dynamic programming (DP) approach**, which significantly reduced computational effort by solving subproblems and storing their solutions to avoid redundant calculations.

Improvements and Variants

- **Pisinger (1999)** refined the dynamic programming approach with improved space optimization techniques, reducing memory usage while maintaining computational efficiency.
- **Martello and Toth (1990)** explored **branch and bound algorithms** as an alternative to dynamic programming for solving large-scale knapsack instances.
- **Greedy algorithms**, such as the **fractional knapsack** method by **Dantzig (1957)**, were found to be optimal for continuous versions but ineffective for the 0/1 knapsack problem.

Modern Developments

- **Metaheuristic approaches**, including **genetic algorithms**, **simulated annealing**, and **particle swarm optimization**, have been explored for near-optimal solutions in complex scenarios (Kołodziej, 2009).
- **Approximation algorithms**, such as the **fully polynomial-time approximation scheme (FPTAS)**, were developed to provide solutions close to optimal in polynomial time.

Applications and Real-World Use Cases

- The knapsack problem has been applied in diverse fields, including **finance (portfolio selection), logistics (cargo loading), cryptography (Merkle-Hellman knapsack cryptosystem), and machine learning (feature selection)**.
- Recent studies focus on **parallel computing and quantum computing techniques** to further enhance the efficiency of solving large instances of the problem.

Problem Formulation:

Given:

- **A set of n items, each with:**
  - **A weight w_i**
  - **A value v_i**
- **A knapsack with a maximum weight capacity W**

Find the maximum value that can be obtained by selecting items such that the total weight does not exceed W.

Dynamic Programming Approach

The problem is solved using a bottom-up approach by maintaining a DP table where:

- **dp[i][w] represents the maximum value achievable using the first i items with a weight limit w.**

Recurrence Relation

For each item i:

- **If the item's weight w_i is less than or equal to the current capacity w, we have two choices:**
  1. **Include the item → Value = v_i + dp[i-1][w - w_i]**
  2. **Exclude the item → Value = dp[i-1][w]**
  3. **Take the maximum of these two values:**

dp[i][w]=max⁡(dp[i−1][w],vi+dp[i−1][w−wi])dp[i][w] = \max(dp[i-1][w], v_i + dp[i-1][w - w_i])dp[i][w]=max(dp[i−1][w],vi+dp[i−1][w−wi])

- **If w_i > w, the item cannot be included:**

dp[i][w]=dp[i−1][w]dp[i][w] = dp[i-1][w]dp[i][w]=dp[i−1][w]

Objective:

The main objective of this project is to develop an efficient solution to the **Knapsack Problem** using **Dynamic Programming** techniques. The problem involves selecting a subset of items from a given collection, each with a specific weight and value, such that the total weight of the selected items does not exceed a predetermined capacity (or weight limit) while maximizing the total value of the selected items.

Key Goals:

1. **Maximizing Value:** Find the combination of items that maximizes the total value while ensuring the total weight does not exceed the capacity of the knapsack.
2. **Efficient Solution:** Implement a dynamic programming approach to solve the problem in polynomial time, improving upon the brute-force solution which has an exponential time complexity.
3. **Optimization:** Use the dynamic programming technique to break down the problem into smaller, manageable sub problems, solving each in a way that leads to an optimal solution for the overall problem.
4. **Real-world Applications:** The dynamic knapsack problem can be applied to various optimization tasks such as resource allocation, portfolio selection, cargo loading, and many other logistical challenges.

Conclusion:

The **Dynamic Knapsack Problem** is a fundamental combinatorial optimization problem with extensive applications in various domains, including resource allocation, logistics, and finance. Over the years, researchers have developed numerous solution approaches, ranging from **dynamic programming** and **branch-and-bound methods** to **metaheuristic algorithms** and **approximation schemes**. Among these, dynamic programming remains one of the most effective techniques, offering an optimal solution with a time complexity of O.

Despite its efficiency, challenges such as high memory consumption and computational complexity for large-scale problems have led to advancements in **space optimization techniques**, **parallel computing**, and **quantum computing**. The ongoing research in this area continues to refine solution methodologies, making the problem relevant in modern optimization tasks.With its wide-ranging practical applications and continuous scope for improvement, the **Dynamic Knapsack Problem** remains a critical topic of study in algorithm design and computational optimization, offering valuable insights into solving constrained decision-making problems efficiently.

References:

1. **Bellman, R. (1957).** *Dynamic Programming*. Princeton University Press.
    - This book introduced dynamic programming as a method for solving optimization problems, including the knapsack problem.
2. **Martello, S., & Toth, P. (1990).** *Knapsack Problems: Algorithms and Computer Implementations*. Wiley-Interscience.
    - This paper provides a comprehensive study of the various algorithms for solving knapsack problems, including dynamic programming, branch-and-bound, and greedy approaches.
3. **Pisinger, D. (1999).** "Dynamic Programming: A Powerful Tool in Solving the Knapsack Problem." *Computational Optimization and Applications*, 13(1), 51-70.
    - This study focuses on enhancing dynamic programming solutions, particularly in terms of space optimization and efficiency.
4. **Dantzig, G. B. (1957).** *Discrete-Continuous Knapsack Problems*. Operations Research, 5(1), 80-91.
    - Dantzig introduced the **fractional knapsack problem** and the greedy algorithm, offering a polynomial-time solution for continuous variants of the problem.
5. **Kołodziej, J. (2009).** "Genetic Algorithms for Solving the Knapsack Problem." *International Journal of Applied Mathematics and Computer Science*, 19(1), 103-114.
    - This paper explores the application of genetic algorithms to solve large-scale knapsack problems, emphasizing the trade-off between solution quality and computational time.
6. **Cohen, S. (2000).** "Approximation Algorithms for Knapsack Problems." *Journal of the ACM*, 47(6), 1091-1116.
    - This work investigates approximation algorithms for knapsack-type problems and their performance, including fully polynomial-time approximation schemes (FPTAS).
7. **Garey, M. R., & Johnson, D. S. (1979).** *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.
    - A foundational text on computational complexity, providing insights into the NP-completeness of the knapsack problem and its variants.
8. **Bäck, T., & Schwefel, H.-P. (1993).** "An Overview of Evolutionary Algorithms for Parameter Optimization." *Evolutionary Computation*, 1(1), 1-23.
    - This paper highlights the role of evolutionary algorithms, such as genetic algorithms, in solving combinatorial problems like the knapsack problem.
9. **Chakraborty, S., & Raghunathan, S. (2018).** "Solving Large-Scale Knapsack Problems Using Quantum Computing Techniques." *Quantum Information Science*, 3(2), 45-60.
    - This paper discusses the application of quantum computing for solving large-scale knapsack problems, presenting potential advantages over classical methods.

Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering