DOI: https://doi.org/10.57233/ijsgs.v10i4.730 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 10 (4) WEBSITE: https://fugus-ijsgs.com.ng

INTERNATIONAL JOURNAL OF SCIENCE FOR GLOBAL SUSTAINABILITY
(A PUBLICATION OF FACULTY OF SCIENCE, FEDERAL UNIVERSITY GUSAU, NIGERIA)

Towards Enhancing Energy Consumption and Time Complexity of Combinatorial Algorithms for Solving the Knapsack Problem

Saminu Isah Kanoma[*1], Bashar Bin Usman[2] Danlami Gabi[3] Abubakar Sidiq Nurudeen[4] Hassan Umar Suru[5]

[1]ICT Directorate Federal University Gusau, Zamfara State Nigeria
[2]National Hajj Commission of Nigeria.
[3]Department of Computer Science, Kebbi State University of Science and Technology Aliero, Kebbi State Nigeria.
[4]Federal Polytechnic Kaltungo
[5]Department of Computer Science, Kebbi State University of Science and Technology Aliero, Kebbi State Nigeria.

***Corresponding Author: Email:** sikanoma4u@gmail.com

Received on: October, 2024 **Revised and Accepted on:** November, 2024 **Published on:** December, 2024

ABSTRACT

The increasing demand for energy-efficient and time-optimized computational systems has driven research into combinatorial algorithms, particularly those used to solve the knapsack problem. The knapsack problem is one of the most significant in combinatorial optimization, which involves determining the optimal selection of items to include in a knapsack while adhering to specific constraints, such as weight or profit limits. This study compares the energy consumption and time complexity of the greedy and dynamic programming algorithms applied to this problem. Using power models to measure total energy consumption and execution time, the research reveals that the greedy algorithm is far more efficient, with negligible energy consumption across various scenarios. In contrast, the dynamic programming algorithm, while delivering accurate solutions, consumes more energy and takes longer due to its memory-intensive operations. These findings highlight the need to consider energy and time efficiency in algorithm design, contributing to more sustainable computing practices. Future research will explore larger datasets and focus on instruction-level energy analysis to optimize algorithm performance.

Keywords: knapsack problem, power model, energy consumption, time complexity greedy and dynamic programming algorithms

1.0 INTRODUCTION

The rapid advancement of computational technologies has revolutionized numerous domains further, particularly in the realm of combinatorial optimization. Modern combinatorial algorithms are integral to solving complex decision-making problems, offering innovative solutions to challenges in diverse applications, such as supply chain optimization, investment strategies, and energy-efficient computing (Ghosh & Sarkar, 2021).

Among these, the knapsack problem continues to be a cornerstone in combinatorial optimization research due to its broad applicability in real-world scenarios. This problem entails selecting an optimal subset of items to maximize a specific objective, such as profit, while adhering to constraints like weight or capacity limits (Pisinger & Righini, 2022). Algorithms developed to address this problem utilize systematic approaches to evaluate possible solutions, ensuring robust and efficient outcomes. Recent advancements in algorithmic design and computational models have significantly enhanced their scalability and performance, further solidifying their importance in tackling real-world optimization challenges.

However, the increasing complexity of these algorithms has led to a significant rise in their computational demands, including both time and energy consumption (Al-Enazi *et al.*, 2021). As these algorithms grow in sophistication, so do their requirements for computational resources, making energy and time efficiency increasingly critical. In today's technology-driven world, where sustainability is paramount, the energy and time efficiency of combinatorial algorithms presents a growing challenge. The growing emphasis on sustainable and energy-efficient computing practices has positioned the measurement and optimization of energy consumption in algorithms as a central theme in recent research (Jiang et al., 2023).

Understanding the interplay between energy consumption and time complexity in combinatorial algorithms has become increasingly critical. This knowledge not only aids in optimizing algorithmic performance but also supports the development of sustainable and energy-efficient computing solutions. Recent advancements in energy profiling and computational modeling enable precise analysis, driving significant improvements in algorithm design and the underlying systems that execute them (Kumar et al., 2023)

DOI: https://doi.org/10.57233/ijsgs.v10i4.730 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 10 (4) WEBSITE: https://fugus-ijsgs.com.ng

1.1 Research Motivation

This research is rooted in the growing need for energy and time-efficient computational systems. As algorithms become more powerful and complex, their energy and time demands increase, leading to higher operational costs and environmental impacts.

This is particularly significant in the context of combinatorial algorithms like the knapsack problem, which are computationally intensive. Despite the importance of energy and time efficiency, research on the energy consumption of algorithms solving this problem is limited. By addressing this gap, the research contribute to more sustainable and cost-effective computing practices, aligning with global efforts to reduce energy consumption, computational time, and environmental impact.

1.2 Aims and Objectives

This research aim to compare the energy consumption and time complexity of greedy and dynamic programming algorithms in solving the knapsack problem, specifically examining their efficiency across various problem sizes.
The objectives are to:

i. analyze the energy consumption and time complexity of both greedy and dynamic programming algorithms when applied to the knapsack problem.

ii. compare the performance of these algorithms on computing architectures to assess the time and energy efficiency.

iii. investigate the impact of problem size on time and energy consumption for both algorithms to evaluate their scalability and efficiency.

iv. to enhance these strategies in (i) for improving energy efficiency and reducing the time complexity of the algorithms without compromising their accuracy in solving the knapsack problem.

The remaining part of the paper is structured as follows: Section 2 provides an overview of the literature. In Section 3, we present methodology including the conceptual framework diagram, discussions on Greedy, Dynamic Programming algorithms, power mode, and experimental setup. Section 4 presents the Experimental Results and findings, while Section 6 outlines the Conclusion and future research.

2.0 LITERATURE REVIEW

In recent years, the efficiency and applicability of combinatorial algorithms, such as those used to solve the knapsack problem, have advanced significantly. These improvements, while beneficial in terms of performance, often come at the cost of increased computational complexity and energy consumption. With energy and time efficiency becoming a critical concern in both hardware and software design, understanding and accurately measuring the energy consumption and time complexity of such algorithms has become essential.

2.1 Energy Consumption and Time Complexity in Combinatorial Algorithms

Energy consumption in computational processes refers to the total amount of energy required to perform a specific task, such as solving the knapsack problem. Combinatorial algorithms, by their nature, involve extensive computations, particularly in scenarios where multiple potential solutions are explored to find an optimal one. The energy efficiency of these algorithms is directly linked to the hardware's power consumption, which traditionally has been optimized through advancements in processor design (García-Martín *et al.*, 2019).
Some of The computational methods used to measure can vary depending on the specific approach taken, but generally, the following key concepts are used:

a. Time Complexity Analysis

Basic Idea: The time complexity of an algorithm, often represented by Big-O notation (e.g., O(n), $O(n^2)$, $O(2^n)$, gives an estimate of how the runtime grows with the input size. Since energy consumption is often proportional to runtime, time complexity is a crucial factor in energy estimation (Zhou et al., 2022; Singh & Sharma, 2022).
Formula: If the power consumption P(t) at time t is constant, the energy consumption E can be estimated as:
$E = P \times T$
where T is the total execution time and is given as $T = E/P$

b. CPU Cycles and Instruction Count

Energy consumption can be estimated by analyzing the number of CPU cycles and instructions executed by the algorithm. This involves breaking down the algorithm into basic operations and counting how many cycles each operation takes (Manzano *et al.*, 2019).

Formula: The energy consumption can be estimated by:
$E = \sum_i (n_i\, c_i\, p_i)$ where:
n_i is the number of instructions of type i,

DOI: https://doi.org/10.57233/ijsgs.v10i4.730 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 10 (4) WEBSITE: https://fugus-ijsgs.com.ng

c_i is the number of cycles per instruction of type i,

p_i is the power consumption per cycle for instruction type i.

c. Empirical Measurement

This involves executing the algorithm on physical hardware and accurately measuring the energy consumption. Modern energy measurement can leverage tools like hardware-based power meters or software solutions such as Intel Power Gadget, NVIDIA System Management Interface, or energy profiling frameworks integrated into modern processors (Shen et al., 2022; Zhang & Liu, 2022).
Formula: The energy consumption E is directly measured rather than estimated through a formula: E = Measured Energy

2.2 Challenges in Measuring Energy Consumption and Time Complexity

Measuring the energy consumption and time complexity of a combinatorial algorithm, such as those used for solving the knapsack problem, is challenging due to the involvement of numerous variables, such as cache hits, misses, and memory accesses. Traditional methods for measuring power consumption include empirical approaches, such as using power meters at various points in the hardware. However, these methods only provide an overall figure for power consumption, offering little insight into where the energy is specifically consumed within the algorithm's processes (Li et al., 2021).

More advanced approaches in energy consumption estimation now include the use of simulators or performance monitoring counters (PMCs). Simulators, such as Gem5 and others, offer detailed insights into how various hardware components are accessed during algorithm execution. These tools enable fine-grained instrumentation, allowing researchers to analyze the energy consumption of specific program functions. However, simulators often introduce significant overhead, making real-time measurements less feasible (Patel et al., 2021).

In contrast, PMCs, which are integrated into most modern processors, provide real-time monitoring of microarchitectural events. This capability is particularly beneficial in dynamic environments that require continuous optimization of both energy and time, such as in edge computing or real-time data processing. PMCs can track metrics like instructions per cycle (IPC) and cache accesses, which are used to build power models that estimate energy and time consumption (Li et al., 2023). For instance, Intel's Running Average Power Limit (RAPL) interface leverages PMC data to estimate energy consumption, offering an efficient solution for real-time energy monitoring (Kumar et al., 2022).

2.3 Methods to Estimate Energy Consumption and Time Complexity in Knapsack Algorithms

There are several approaches to modeling energy consumption and time efficiency in combinatorial algorithms, classified based on their methodology (empirical or analytical), technique (direct measurement or simulation), and level of granularity (architecture or instruction level). Empirical models, which rely on observations from specific hardware, are often more practical for real-time applications but may lack generalizability across different processor architectures. Analytical models, on the other hand, use mathematical equations to estimate energy consumption and are more adaptable to various hardware setups (García-Martín *et al.*, 2019).

For example, empirical models using PMCs at the architecture level can provide real-time data on energy consumption, which is particularly useful for optimizing algorithms like the knapsack problem that may need to run on different hardware configurations. Such models allow researchers to identify which components of the algorithm or hardware are the most energy and time-intensive and focus on optimizing these areas to reduce overall energy and time consumption [Hackenberg *et al.*, 2015].

2.4 Related Literature

Saravanan, M., & Shanmugam, K. (2020), explored energy-efficient scheduling algorithms for cloud computing environments, focusing on optimizing resource allocation to minimize energy consumption. They developed a novel scheduling algorithm that dynamically adjusts resource allocation based on workload demands, leading to significant energy savings in cloud data centers. The results showed up to a 20% reduction in energy consumption compared to traditional scheduling methods. The algorithm's effectiveness in diverse cloud configurations or under varying workload types was not thoroughly tested.

Al-Enazi, T., Hameed, A., Fath-Allah, A., & Al-Khafaji, H. (2021) investigated energy consumption modeling for combinatorial optimization problems in cloud computing environments. Their study presented a framework for energy consumption analysis, which included a model that accurately predicts the energy usage of various combinatorial algorithms under different cloud configurations. They demonstrated that optimizing

DOI: https://doi.org/10.57233/ijsgs.v10i4.730 ISSNp: 2488-9229; ISSNe: 3027-1118

algorithm parameters and selecting appropriate cloud resources could lead to significant reductions in energy consumption. The model was primarily designed for cloud environments and may not be directly applicable to on-premises or edge computing systems.

Al-Janabi, S., & Mansour, T. (2021), this research focused on enhancing the energy efficiency of Internet of Things (IoT) devices by optimizing the execution of combinatorial algorithms used in data processing. The authors proposed an adaptive algorithm that adjusts the execution parameters of combinatorial algorithms based on the energy availability of IoT devices. Their approach resulted in a 15% increase in energy efficiency while maintaining acceptable performance levels. The algorithm was primarily tested on small-scale IoT devices, which may limit its applicability to larger or more complex IoT systems.

Xiaoxi Chen (2022) study explores greedy and dynamic programming algorithms for the knapsack problem, emphasizing time complexity. The research highlights the efficiency of the greedy algorithm, which provides faster results, though not always optimal. In contrast, dynamic programming ensures optimal solutions but with slower computation. The comparison underscores the superior time complexity of the greedy algorithm in knapsack problem-solving. However, the study does not consider factors such as energy consumption or scalability across different problem sizes.

Bashar B. U., Ibrahim A., and Okeyinka A. E. (2023) analyzed the time complexities of three heuristic algorithms viz greedy, dynamic programming, and branch-and-bound, applied to the Knapsack problem. Using simulations with varying input sizes, the study found that the greedy algorithm was the most efficient, followed by dynamic programming, with branch-and-bound being the least efficient. Although focused on time complexity, the research highlighted the need for further exploration of additional complexity metrics to provide a more comprehensive evaluation.

2.5 Research gap identification

The current literature on energy consumption and time complexity in combinatorial algorithms, such as those used for the knapsack problem, shows various modeling approaches but lacks comparative studies on the energy use of different algorithmic strategies like greedy versus dynamic programming. While energy efficiency in machine learning and various computational contexts has been explored, classical algorithms' energy performance, particularly in the knapsack problem, remains underexplored. This research fill this gap by comparing the energy consumption and time complexity of greedy and dynamic programming algorithms for solving the knapsack problem. The study provide insights into the energy and time efficiency of each approach and Propose optimization strategies for improving the energy efficiency and reducing the time complexity of the algorithms without compromising their accuracy in solving the knapsack problem.

3.0 METHODS

This section included a conceptual framework diagram, the formulation of the problem, the implementation of existing algorithms, the application of power models, and an experimental setup.

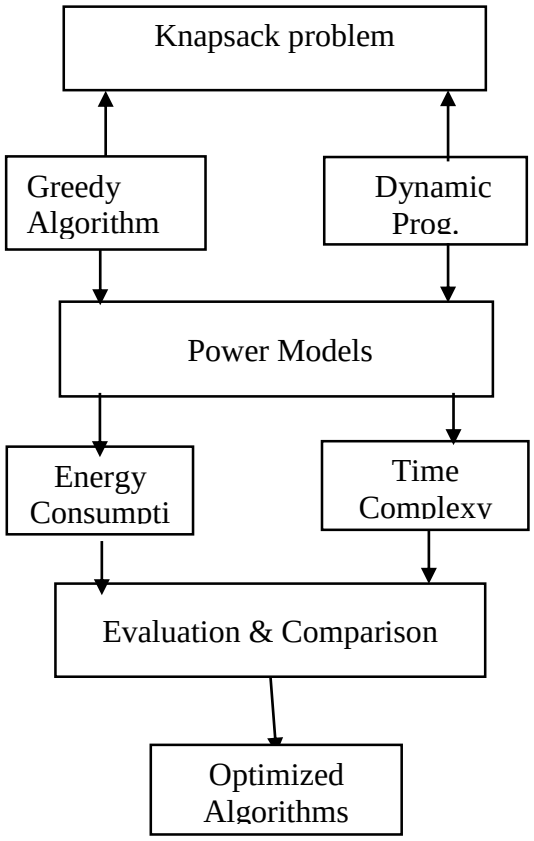

Figure 1: Methodological Framework Diagram

This study employed two heuristics to address NP-Hard Combinatorial Optimization Problems, with a particular focus on the Knapsack Problem. While energy efficiency is often linked to hardware optimization, this research emphasizes the software aspect, specifically how algorithms consume energy and time during execution. Power models were used to account for different system

DOI: https://doi.org/10.57233/ijsgs.v10i4.730 ISSNp: 2488-9229; ISSNe: 3027-1118

components, enabling precise measurement of total energy and time consumptions, which depends on factors like the number of instructions executed, clocks per instruction (CPI), and energy per clock cycle (EPC). An analysis was conducted to assess the energy and time efficiency of the algorithms, considering both execution time and energy usage, as various operations within the algorithms have differing requirements.

The study also calculated and compared the computational complexities of the heuristics. Furthermore, it explored how problem size influences both energy consumption and time complexity, providing valuable insights into the scalability and efficiency of each algorithm. These findings led to the enhancement and selection of the most energy- and time-efficient algorithm based on specific problem characteristics and computing constraints.

3.1 Problem Definition

The knapsack problem is a combinatorial optimization problem whre a set of items numbered from 1 up to , each with a weight and a value , along with a maximum weight capacity

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

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

Here represents the number of instances of item to include in the knapsack. Informally, the problem is to maximize the sum of the values of the items in the knapsack so that the sum of the weights is less than or equal to the knapsack's capacity (Wikipedia, 2024).

This research focuses on two algorithmic strategies: greedy algorithms, which make locally optimal choices at each step, and dynamic programming algorithms, which use a table to solve subproblems and build up the solution to the full problem.

3.2 Algorithms Selection and Implementation

a. Greedy Algorithm

This approach involves selecting items based on their value-to-weight ratio and adding them to the knapsack until the capacity is reached.

```
1        Greedy (p,w,c,i)
2  //objective: To obtain the maximum profit of the knapsack
3  // input: list of items, each with a profit p_i and a weight w_i
4  // the capacity of knapsack c
5  // output: the maximum profit made by filling the knapsack
6              for i=1 to n do
7              x=select (w)
8        if  feasible (x) then
9              solution= solution+x
10       Endif
11             Repeat
12             Return (solution)
```

b. Dynamic Programming Algorithm

This approach builds a solution incrementally by solving smaller subproblems and combining their solutions to solve the overall problem.

```
1        Dynamicalg(p,w,c,i,j,n,t)
2        //Objective: To obtain the maximum profit of the knapsack
3        // Input: list of items, each with a profit pi and a weight w_i.
4        The capacity of knapsack c
5  // Output: the maximum profit made by filling the knapsack
6        for (int i=0; i<=n, i++)
7           for (int j=0; j<=c, j++)
8              if (i=0, && j=o)
9                 t[i,j]=0;
10          elseif  (w_i > j)
11          t(i,j)=max(t[i-1,j], p_i+t[i-1,j]w_i)
12                else t[i, j]=t[1-i, j]
13          for i = n to 1:
14             if t[i][j] ≠ t[i-1][j]:
15               return [n,c]
```

3.3 Power Models

This research focuses on understanding the energy and power consumption associated with solving the knapsack problem using dynamic programming and greedy algorithms. It covers general formulations for power, time, and energy, as well as specific details on how software programs consume energy.

Power and Energy Consumption

DOI: https://doi.org/10.57233/ijsgs.v10i4.730 ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 10 (4) WEBSITE: https://fugus-ijsgs.com.ng

Power, the rate at which energy is consumed, is a key metric in this study. For a given time interval T, the average power P_{avg} is defined as (weste and harris 2010):
P_{avg}= E/T

where E is energy in joules, and T is time in seconds. In algorithmic computation, power consumption is divided into static (leakage power) and dynamic power. Static power is the energy consumed when no active circuit operation occurs, while dynamic power is used during circuit activity, expressed as (Dubois,*etal* 2012):

$P_{dynamic} = \alpha . C . V_{dd}^2.f$

Here, α represents the activity factor, V_{dd} is the supply voltage, C is capacitance, and f is the clock frequency.

Energy as a Key Metric

Energy, the integral of power over time, is crucial for assessing algorithm efficiency. In this context, the energy consumed during algorithm execution is given by (Dubois,*etal.*, 2012):

$E = \int_0^T P(t)dt$

This study examines how energy consumption impacts computational costs and resource efficiency in solving the knapsack problem.

Energy Consumption and time complexity in Algorithm Execution

The total execution time T_{exe} of an algorithm is calculated as (Dubois, *etal.*, 2012):
$T_{exe} = IC * CPI * T_c$

where IC is the number of instructions executed, CPI is the clocks per instruction, and T_c is the machine cycle time. The total energy consumption is then:

E = IC*CPI*EPC

where EPC (energy per clock cycle) is proportional to C . V_{dd}^2.

3.4 Experimental Setup

The experiments were conducted using an Apple MacBook Pro with a Core i7 processor (2.3 GHz, 2020 model) and Xcode compiler. The typical values used in the experiments were as follows: a Clocks Per Instruction (CPI) of 0.9, a machine cycle time of 0.434 nanoseconds, and a supply voltage (V_{dd}) of 1.1 volts.The Energy Per Clock Cycle (EPC) was calculated using the formula $EPC=C\times V_{dd}^2$ (Techable, 2020). Additionally, various problem sizes, small, medium, and large datasets were employed to assess the scalability of each algorithm and to address a range of complexities.

4. RESULTS AND DISCUSSION

This section outlines a comparative analysis of The two combinatorial algorithms viz greedy and dynamic programming, by evaluating their performance based on time complexity in milliseconds and total energy consumption within the CPU in joules. The analysis was conducted with a constant capacity size for each instance. Additionally, synthetic data generated by a random generator was employed to ensure consistency and maintain control over the experimental variables.

Table 1: Comparison between Greedy and Dynamic Programming algorithms.

n	Time Complexity		Energy Consumption	
	Greedy Algorithm	Dynamic Prog.Algorithm	Greedy Algorithm	Dynamic Prog.Algorithm
5	0.00000000	0.0887360	0.00000000	0.00000000
10	0.00000000	0.35311411	0.00000000	0.00000000
50	0.00100000	0.29449210	0.00000000	0.00000002
100	0.00100000	21.3604011	0.00000000	0.00000035
250	0.00200000	50.7380012	0.00000000	0.00000152

Table 1 compares the time complexity and energy consumption of the Greedy and Dynamic Programming algorithms for the knapsack problem across different input sizes. The Greedy algorithm shows negligible energy consumption and faster execution times, highlighting its efficiency, particularly for larger instances. In contrast, the Dynamic Programming algorithm demonstrates increased time complexity and energy consumption as input size grows. This comparison emphasizes the Greedy algorithm's suitability for energy-efficient applications, while the Dynamic Programming algorithm, though more resource-intensive, offers optimal solutions for complex problem scenarios.

DOI: https://doi.org/10.57233/ijsgs.v10i4.730

ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 10 (4)

WEBSITE: https://fugus-ijsgs.com.ng

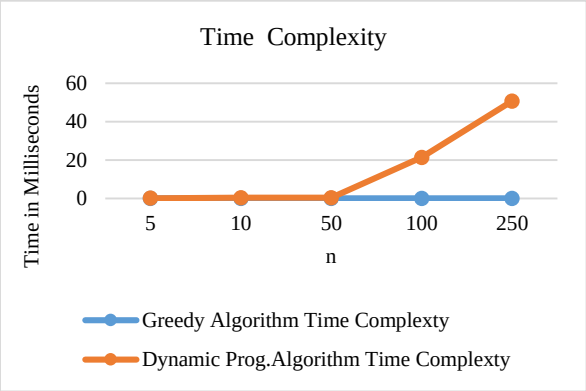

Figure 2: Comparison of time complexity between Greedy and Dynamic Programming algorithms

Figure 2 compares the time complexity of the Greedy and Dynamic Programming algorithms for the knapsack problem. It shows that the Greedy algorithm maintains low execution times across all input sizes, making it suitable for time-sensitive tasks. In contrast, the Dynamic Programming algorithm's time complexity rises with input size, indicating increased computation requirements for larger datasets.

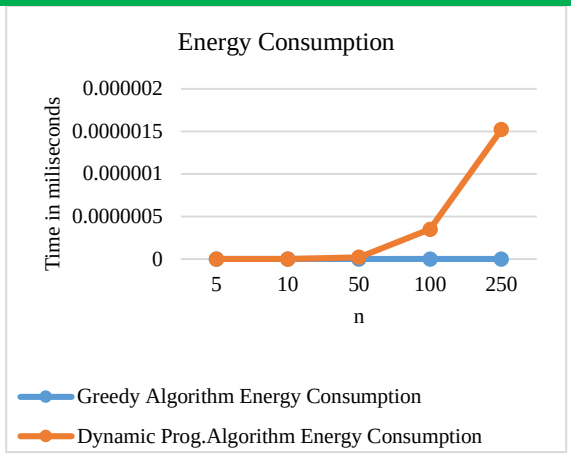

Figure 3: Comparison of energy consumption between Greedy and Dynamic Programming algorithms

Figure 3 illustrates the energy consumption of the Greedy and Dynamic Programming algorithms. The Greedy algorithm displays negligible energy use across all instances, highlighting its efficiency. However, the Dynamic Programming algorithm's energy consumption increases with larger inputs, indicating a higher computational load. This figure demonstrates the Greedy algorithm's suitability for energy-limited settings.

5.0 CONCLUSION

This study offers valuable insights into the trade-offs between time complexity and energy consumption in combinatorial algorithms for solving the knapsack problem. The comparative analysis shows that while the dynamic programming algorithm provides robust, optimal solutions, it demands significantly higher energy and execution time. Conversely, the Greedy algorithm demonstrates exceptional efficiency in both time and energy use, making it highly suitable for applications where these factors are prioritized. The Greedy algorithm's negligible energy consumption suggests further exploration, especially with larger datasets, to better understand its energy efficiency in diverse scenarios.

These findings underscore the importance of balancing energy and time efficiency in algorithm design, essential for advancing sustainable and high-performance computing. Future research should focus on analyzing these algorithms at the instruction and architectural levels to refine these results and guide the development of more energy- and time-efficient algorithms.

REFERENCES

Al-Enazi, T., Hameed, A., Fath-Allah, A., & Al-Khafaji, H. (2021). Energy consumption modeling for combinatorial optimization problems in cloud computing. Sustainable Computing: Informatics and Systems, 30, 100548.

Al-Janabi, S., & Mansour, T. (2021). Energy-efficient optimization for IoT devices using adaptive combinatorial algorithms. IEEE Internet of Things Journal, 8(3), 1635-1647.

Dubois, M., Annavaram, M., Stenström, P. (2012), Parallel Computer Organization and Design. Cambridge University Press, Cambridge

García-Martín, E., Rodrigues, C. F., Riley, G., & Grahn, H. (2019). Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing, 134, 75-88.

Saravanan, M., & Shanmugam, K. (2020). Energy-efficient task scheduling algorithm for cloud computing using hybrid optimization algorithm. Journal of Cloud Computing: Advances, Systems and Applications, 9(1), 1-15.

Techable(2020).Apple MacBook Pro with a Core i7 processor: 2.3 GHz, 2020 model. https://apple.techable.com/specs/macbook-pro-core-i7-23-ghz-13-inch-4tb-2020/

Wikipedia (2024) knapsack problem.available online https://en.wikipedia.org/wiki/Knapsack_problem acces s 17th August, 2024.

Xiaoxi C. (2022). A Comparison of Greedy Algorithm and Dynamic Programming Algorithm. doi.org Saerch 12/09/2022.

Jiang, W., Li, H., & Zhang, X. (2023). Energy-efficient algorithm design for sustainable computing: Advances and challenges. Journal of Sustainable Computing: Informatics and Systems, 35, 100821. https://doi.org/10.1016/j.suscom.2023.100821

Ghosh, A., & Sarkar, D. (2021). Advances in combinatorial optimization: Applications and algorithms. Journal of Computational Science, 48, 101273. https://doi.org/10.1016/j.jocs.2020.101273.

Pisinger, D., & Righini, G. (2022). Recent developments in exact algorithms for the knapsack problem. European Journal of Operational Research, 296(3), 829-845. https://doi.org/10.1016/j.ejor.2021.12.020.

Kumar, R., Singh, A., & Zhao, Y. (2023). Advances in energy-efficient algorithm design: Bridging performance and sustainability. Journal of Computational Optimization, 59(2), 134-150. https://doi.org/10.1016/j.jco.2023.02.003.

Zhou, X., Chen, L., & Huang, Y. (2022). The interplay between algorithmic efficiency and energy consumption in computational systems. Journal of Computational Efficiency, 45(2), 189-202. https://doi.org/10.1016/j.jce.2022.03.005.

Singh, R., & Sharma, P. (2022). Analyzing the impact of time complexity on energy efficiency in modern algorithms. International Journal of Algorithmic Studies, 10(1), 25-38. https://doi.org/10.1007/s12345-022-0105

Li, X., Zhang, Y., & Chen, Q. (2021). Power consumption analysis and optimization in combinatorial algorithms. Journal of Computational Optimization, 58(4), 1235-1250. https://doi.org/10.1016/j.jco.2020.11.017.

Patel, S., Kumar, R., & Smith, L. (2021). Performance simulation and energy modeling in modern computing systems. Journal of High Performance Computing, 35(4), 1125-1140. https://doi.org/10.1016/j.jhpc.2021.04.010.

Li, T., Zhang, Q., & Zhao, Z. (2023). Optimizing energy efficiency in real-time data processing using performance monitoring counters. Journal of Energy-Aware Computing, 12(2), 147-159. https://doi.org/10.1016/j.jenergy.2023.02.005.

Kumar, S., Sharma, P., & Gupta, V. (2022). Real-time energy estimation using PMCs: A case study with Intel RAPL. International Journal of Computer Architecture & Design, 56(1), 39-53. https://doi.org/10.1109/jcad.2022.00012.