Penerapan Program Dinamis dalam Penyelesaian Masalah Ransum Optimal untuk Hewan Ternak

Maggie Zeta Rosida Simangunsong - 13521117
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521117@std.stei.itb.ac.id

Abstract*—Makalah ini membahas penerapan algoritma program dinamis dalam penyelesaian masalah ransum optimal untuk hewan ternak. Masalah ini melibatkan pemilihan kombinasi pakan yang memenuhi kebutuhan gizi hewan ternak dengan biaya minimal. Kontribusi utama makalah ini adalah penerapan algoritma program dinamis dan analisis kinerjanya dalam mencari solusi ransum optimal, serta eksperimen untuk menguji keefektifan dan keakuratan algoritma. Algoritma program dinamis digunakan untuk memecahkan masalah secara rekursif dengan memanfaatkan submasalah yang lebih kecil dan menyimpan hasilnya dalam tabel. Eksperimen dilakukan dengan menggunakan dataset yang mencakup informasi nutrisi pakan dan biaya, serta kebutuhan gizi hewan ternak. Pengukuran kinerja meliputi waktu eksekusi dan keakuratan solusi yang ditemukan. Hasil eksperimen menunjukkan bahwa algoritma program dinamis mampu mencari solusi ransum optimal dengan biaya minimal secara efisien. Kelebihan algoritma program dinamis adalah kemampuannya dalam mengatasi overlapping submasalah dan menghindari pengulangan perhitungan yang tidak perlu. Namun, kelemahan algoritma ini terletak pada kompleksitas ruang yang lebih tinggi dibandingkan dengan pendekatan lain. Penelitian ini membuktikan bahwa algoritma program dinamis merupakan pendekatan yang efektif dan dapat diandalkan dalam memecahkan masalah ransum optimal untuk hewan ternak, dengan potensi untuk diterapkan dalam konteks industri peternakan dan pertanian.

Keywords—Program Dinamis, Ransum Optimal, Pemilihan Pakan, Kinerja Algoritma, Eksperimen

I. PENDAHULUAN

Dalam industri peternakan dan pertanian, pemenuhan kebutuhan gizi hewan ternak merupakan faktor krusial untuk memastikan kesehatan dan produktivitas yang optimal. Salah satu aspek penting dalam pemenuhan kebutuhan gizi tersebut adalah pemilihan ransum atau kombinasi pakan yang tepat. Pemilihan ransum yang optimal harus mempertimbangkan ketersediaan pakan, komposisi nutrisi, dan biaya yang efisien. Namun, mencari solusi ransum optimal dalam skala besar dapat menjadi tugas yang kompleks dan memakan waktu.

Makalah ini bertujuan untuk mengaplikasikan algoritma program dinamis dalam penyelesaian masalah ransum optimal untuk hewan ternak. Algoritma program dinamis merupakan pendekatan yang efektif dalam memecahkan masalah optimasi dengan memecahnya menjadi submasalah yang lebih kecil dan menyimpan hasilnya dalam tabel. Dalam konteks ini, algoritma program dinamis akan digunakan untuk memilih kombinasi pakan yang memenuhi kebutuhan gizi hewan ternak dengan biaya minimal.

Makalah ini akan memfokuskan pada pemilihan ransum optimal untuk hewan ternak dengan mempertimbangkan komposisi nutrisi yang dibutuhkan dan biaya yang terkait. Kami akan menganalisis implementasi algoritma program dinamis dalam menyelesaikan masalah ini dan melakukan eksperimen untuk menguji kinerja serta keakuratan solusi yang ditemukan. Meskipun penelitian sebelumnya telah mengusulkan pendekatan lain untuk masalah ransum optimal, seperti metode heuristik atau pendekatan matematis, kami akan menunjukkan bahwa algoritma program dinamis dapat menjadi alternatif yang efisien dan efektif dalam menyelesaikan masalah ini.

II. DASAR TEORI

A. Algoritma Program Dinamis

Algoritma program dinamis adalah teknik pemrograman yang digunakan untuk memecahkan masalah optimasi dengan memecahnya menjadi submasalah yang lebih kecil dan menyimpan hasilnya dalam tabel (atau matriks) untuk menghindari pengulangan perhitungan yang tidak perlu. Berikut ini adalah langkah-langkah umum dalam implementasi algoritma program dinamis:

1. Identifikasi Struktur Submasalah

   Pertama, identifikasi struktur submasalah yang terkait dengan masalah yang ingin diselesaikan. Submasalah adalah masalah yang lebih kecil yang dapat dipecahkan secara terpisah dan berkontribusi pada solusi akhir.

2. Pembentukan Rekurensi

   Setelah mengidentifikasi submasalah, kita perlu menentukan persamaan rekurensi yang menghubungkan submasalah dengan solusi yang lebih kecil. Persamaan rekurensi harus memungkinkan perhitungan solusi submasalah berdasarkan solusi submasalah yang lebih kecil.

3. Tabel Memoization

Gunakan tabel memoization (tabel penyimpanan) untuk menyimpan hasil perhitungan solusi submasalah yang telah dipecahkan. Tabel ini akan menghindari pengulangan perhitungan yang tidak perlu dan meningkatkan efisiensi algoritma. Tabel ini biasanya berupa matriks atau array, di mana setiap sel dalam tabel akan menyimpan solusi submasalah.

4. Pengisian Tabel Memoization

   Isi tabel memoization dengan hasil perhitungan solusi submasalah secara sistematis. Mulai dari submasalah yang paling kecil dan naik ke submasalah yang lebih besar. Pastikan untuk menggunakan hasil perhitungan solusi submasalah yang lebih kecil yang telah tersimpan dalam tabel.

5. Konstruksi Solusi Optimal

   Setelah semua submasalah dipecahkan dan tabel memoization terisi, kita dapat menggunakan tabel memoization untuk membangun solusi optimal. Langkah ini melibatkan melacak kembali melalui tabel memoization dan memilih solusi submasalah yang memaksimalkan (atau meminimalkan, tergantung pada jenis masalah) nilai objektif.

6. Mengembalikan Solusi

   Terakhir, kembalikan solusi optimal yang telah dikonstruksi berdasarkan tabel memoization. Solusi ini akan menjadi solusi optimal untuk masalah awal yang lebih besar.

Dalam implementasi algoritma program dinamis, penting untuk memperhatikan aspek penggunaan memori dan waktu eksekusi. Beberapa teknik optimasi, seperti penggunaan teknik kompresi tabel memoization atau optimasi rekursi, dapat digunakan untuk meningkatkan efisiensi algoritma.

Dengan memahami langkah-langkah ini, kita dapat menerapkan algoritma program dinamis dalam pemecahan masalah ransum optimal untuk hewan ternak dan menganalisis kinerjanya dalam mencari solusi optimal dengan biaya minimal.

B. Masalah Ransum Optimal untuk Hewan Ternak

Masalah Ransum Optimal untuk Hewan Ternak adalah masalah optimasi yang melibatkan pemilihan kombinasi pakan yang memenuhi kebutuhan gizi hewan ternak dengan biaya minimal. Dalam masalah ini, kita harus memilih jenis dan jumlah pakan yang tepat untuk mencapai kebutuhan nutrisi yang diinginkan sambil meminimalkan biaya yang dikeluarkan.

Pada dasarnya, masalah ini melibatkan beberapa variabel penting:

1. Jenis Pakan

   Terdapat berbagai jenis pakan yang tersedia dengan komposisi nutrisi yang berbeda. Setiap jenis pakan memiliki kandungan protein, energi, serat, vitamin, mineral, dan nutrisi lainnya yang berbeda.

2. Kebutuhan Nutrisi Hewan Ternak

   Setiap hewan ternak memiliki kebutuhan nutrisi spesifik yang harus dipenuhi agar dapat tumbuh, berkembang, dan berproduksi dengan baik. Kebutuhan nutrisi ini termasuk kebutuhan protein, energi, vitamin, mineral, dan komponen nutrisi lainnya.

3. Batasan Ketersediaan Pakan

   Terdapat batasan dalam ketersediaan pakan, baik dalam jumlah maupun jenis pakan yang tersedia. Ketersediaan pakan ini dapat tergantung pada faktor ekonomi, musim, dan kondisi lokal.

4. Batasan Biaya

   Selain memenuhi kebutuhan nutrisi, kita juga perlu meminimalkan biaya yang dikeluarkan untuk membeli pakan. Setiap jenis pakan memiliki biaya yang berbeda.

Dalam pemecahan masalah ini, langkah utama adalah mencari kombinasi pakan yang memenuhi kebutuhan nutrisi hewan ternak dengan meminimalkan biaya yang terkait. Ini melibatkan pemodelan masalah dengan menggunakan matriks atau tabel, di mana setiap baris mewakili jenis pakan, dan setiap kolom mewakili komponen nutrisi atau biaya.

Selanjutnya, dengan menggunakan algoritma program dinamis, kita dapat mengisi tabel memoization dengan perhitungan solusi submasalah. Hal ini memungkinkan kita untuk menghindari pengulangan perhitungan dan meningkatkan efisiensi algoritma. Setelah tabel memoization terisi, kita dapat membangun solusi optimal dengan memilih kombinasi pakan yang meminimalkan biaya dan memenuhi kebutuhan nutrisi.

Dengan demikian, masalah ransum optimal untuk hewan ternak merupakan masalah kompleks yang memerlukan perhitungan yang cermat dan strategi pemodelan yang baik untuk mencari solusi yang memadai dalam pemenuhan kebutuhan nutrisi dengan biaya minimal.

C. Pemodelan Masalah Ransum Optimal

Pemodelan Masalah Ransum Optimal melibatkan representasi matematis dari masalah tersebut. Dalam pemodelan ini, beberapa komponen penting yang perlu diperhatikan adalah jenis pakan, kebutuhan nutrisi hewan ternak, batasan ketersediaan pakan, dan batasan biaya.

Berikut ini adalah Langkah-langkah untuk memodelkan Masalah Ransum Optimal:

1. Identifikasi Variabel

   a. Variabel Keputusan

      - Jumlah pakan yang akan diberikan dari setiap jenis pakan.

   b. Variabel Penunjang

      - Biaya setiap jenis pakan.
      - Komposisi nutrisi (protein, energi, serat, vitamin, mineral, dll.) dari setiap jenis pakan.
      - Kebutuhan nutrisi hewan ternak.

2. Fungsi Tujuan
    a. Minimalkan biaya total pakan yang diberikan.
    b. Biaya total dihitung dengan mengalikan jumlah pakan dari setiap jenis dengan biaya masing-masing jenis pakan.
3. Batasan
    a. Ketersediaan pakan
        - Batasan ketersediaan masing-masing jenis pakan.
    b. Kebutuhan nutrisi
        - Kebutuhan nutrisi minimum yang harus dipenuhi untuk setiap komponen nutrisi (protein, energi, serat, vitamin, mineral, dll.).
4. Matriks Nutrisi
    a. Membuat matriks atau tabel yang menggambarkan komposisi nutrisi dari setiap jenis pakan.
    b. Setiap baris pada matriks ini mewakili jenis pakan, sedangkan kolom-kolom merepresentasikan komponen nutrisi (protein, energi, serat, vitamin, mineral, dll.).
5. Matriks Ketersediaan
    a. Membuat matriks atau tabel yang menggambarkan ketersediaan masing-masing jenis pakan.
    b. Setiap baris pada matriks ini mewakili jenis pakan, sedangkan kolom-kolom merepresentasikan batasan ketersediaan.
6. Perumusan Masalah
    a. Memformulasikan fungsi tujuan dan batasan ke dalam bentuk persamaan matematis.
    b. Menentukan batasan ketersediaan pakan dan kebutuhan nutrisi sebagai batasan kesetaraan atau ketidaksamaan.
    c. Menggunakan variabel keputusan dan matriks nutrisi untuk memodelkan persamaan-persamaan nutrisi.
7. Solusi dan Analisis
    a. Menggunakan algoritma program dinamis atau metode optimisasi lainnya untuk mencari solusi optimal.
    b. Solusi optimal akan memberikan kombinasi pakan yang memenuhi kebutuhan nutrisi dengan biaya minimal.

Pemodelan yang baik akan mempertimbangkan kompleksitas nutrisi yang dibutuhkan hewan ternak, ketersediaan dan biaya pakan, serta memperhitungkan batasan yang relevan. Dengan pemodelan yang akurat, kita dapat mencari solusi yang efisien dan mengoptimalkan kebutuhan nutrisi dengan biaya yang minimal untuk hewan ternak kita.

D. Langkah-langkah Algoritma Program Dinamis dalam Penyelesaian Masalah Ransum Optimal

Berikut ini adalah langkah-langkah lebih detail dalam menerapkan algoritma program dinamis untuk menyelesaikan Masalah Ransum Optimal:

1. Identifikasi Submasalah

    Submasalah dalam masalah Ransum Optimal adalah pemilihan jumlah pakan yang optimal untuk setiap jenis pakan dalam mencapai kebutuhan nutrisi dengan biaya minimal.
2. Pembentukan Rekurensi
    a. Tentukan rekurensi yang menghubungkan solusi submasalah yang lebih kecil dengan solusi submasalah yang lebih besar.
    b. Misalnya, rekurensi dapat dinyatakan sebagai berikut:

        - DP[i] = minimum biaya untuk mencapai kebutuhan nutrisi dengan menggunakan i jenis pakan.

        - DP[i] = min(DP[i - 1] + cost[i], DP[i - 2] + cost[i-1], ..., DP[i - k] + cost[i-k+1]), di mana k adalah batasan jumlah jenis pakan yang dapat dipilih.
3. Tabel Memoization
    a. Buat tabel memoization berukuran (n + 1) x (m + 1), di mana n adalah jumlah jenis pakan dan m adalah jumlah kebutuhan nutrisi.
    b. Setiap sel dalam tabel memoization akan menyimpan biaya minimum untuk mencapai kebutuhan nutrisi dengan menggunakan jumlah jenis pakan tertentu.
4. Pengisian Tabel Memoization
    a. Mulai dari submasalah terkecil, yaitu menggunakan 0 jenis pakan.
    b. Isi tabel memoization secara berurutan, dari kiri ke kanan dan dari atas ke bawah, dengan memperhitungkan rekurensi yang telah ditentukan.
    c. Setiap sel dalam tabel diisi dengan biaya minimum untuk mencapai kebutuhan nutrisi dengan menggunakan jenis pakan yang sesuai.
5. Konstruksi Solusi Optimal
    a. Setelah tabel memoization terisi, kita dapat melacak kembali melalui tabel untuk membangun solusi optimal.
    b. Mulai dari sel terakhir di tabel, periksa nilai biaya minimum yang ada.
    c. Dari nilai biaya minimum tersebut, kita dapat menentukan jenis pakan yang digunakan.
    d. Lanjutkan melacak kembali ke sel sebelumnya dengan memperbarui kebutuhan nutrisi dan jenis

pakan yang dipilih hingga mencapai sel awal tabel.

6. Mengembalikan Solusi

   a. Setelah mencapai sel awal tabel, kita akan memiliki kombinasi pakan yang meminimalkan biaya dan memenuhi kebutuhan nutrisi.

   b. Kembalikan solusi optimal yang telah dikonstruksi sebagai hasil akhir dari algoritma.

Dengan langkah-langkah ini, algoritma program dinamis dapat digunakan untuk mencari solusi Ransum Optimal yang memenuhi kebutuhan nutrisi dengan biaya minimal. Penting untuk memperhatikan kompleksitas waktu dan memori algoritma, serta mempertimbangkan penggunaan teknik optimisasi seperti penggunaan teknik kompresi tabel memoization untuk meningkatkan efisiensi algoritma.

III. Implementasi dan Pembahasan

A. *Implementasi Program*

Implementasi dilakukan menggunakan Bahasa pemrograman Python.

```python
def ransum_optimal(pakan, kebutuhan, biaya):
    n = len(pakan)  # Jumlah jenis pakan
    m = len(kebutuhan)  # Jumlah kebutuhan nutrisi

    # Membuat tabel memoization
    dp = [[float('inf')] * (m + 1) for _ in range(n + 1)]

    # Kasus dasar: menggunakan 0 jenis pakan
    for i in range(m + 1):
        dp[0][i] = 0

    # Mengisi tabel memoization secara berurutan
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            # Pilihan 1: Tidak menggunakan pakan ke-i
            dp[i][j] = dp[i - 1][j]

            # Pilihan 2: Menggunakan pakan ke-i
            if j >= kebutuhan[i - 1]:
                dp[i][j] = min(dp[i][j], dp[i][j - kebutuhan[i - 1]] + biaya[i - 1])

    # Konstruksi solusi optimal
    solusi = []
    i = n
    j = m
    while i > 0 and j > 0:
        if dp[i][j] != dp[i - 1][j]:
            solusi.append(pakan[i - 1])
            j -= kebutuhan[i - 1]
```

Gambar 3.1 Implementasi Kode Program

```python
        i -= 1

    solusi.reverse()  # Membalik urutan solusi

    # Mengembalikan solusi optimal
    return solusi


# Input pengguna
def get_input():
    pakan = input("Masukkan jenis pakan (pisahkan dengan koma): ").split(", ")
    kebutuhan = list(map(int, input("Masukkan kebutuhan nutrisi (pisahkan dengan koma): ").split(", ")))
    biaya = list(map(int, input("Masukkan biaya pakan (pisahkan dengan koma): ").split(", ")))
    return pakan, kebutuhan, biaya


# Pengujian program
if __name__ == "__main__":
    pakan, kebutuhan, biaya = get_input()

    solusi_optimal = ransum_optimal(pakan, kebutuhan, biaya)
    if solusi_optimal:
        print("Jenis pakan yang dipilih untuk ransum optimal:")
        for p in solusi_optimal:
            print(p)
    else:
        print("Tidak ada solusi optimal yang ditemukan.")
```

Gambar 3.2 Implementasi Kode Program

Implementasi kode di atas adalah untuk menyelesaikan masalah pemilihan ransum optimal menggunakan algoritma program dinamis. Berikut adalah penjelasan detail implementasi kode tersebut:

1. Fungsi `ransum_optimal(pakan, kebutuhan, biaya)`: Fungsi ini menerima tiga parameter, yaitu `pakan` (daftar jenis pakan yang tersedia), `kebutuhan` (daftar kebutuhan nutrisi), dan `biaya` (daftar biaya untuk setiap jenis pakan). Fungsi ini mengembalikan solusi optimal dalam bentuk daftar jenis pakan yang dipilih.

a. `n = len(pakan)`: Variabel `n` menyimpan jumlah jenis pakan.

b. `m = len(kebutuhan)`: Variabel `m` menyimpan jumlah kebutuhan nutrisi.

c. `dp = [[float('inf')] * (m + 1) for _ in range(n + 1)]`: Membuat tabel memoization dengan ukuran `(n + 1) x (m + 1)`. Awalnya, setiap sel diisi dengan nilai tak terhingga (`inf`).

d. Kasus dasar: Mengeset nilai `dp[0][i] = 0` untuk setiap `i` dari 0 hingga `m`. Ini menunjukkan bahwa jika tidak ada jenis pakan yang digunakan, biaya yang diperlukan adalah 0.

e. Mengisi tabel memoization secara berurutan: Digunakan nested loop untuk mengisi tabel `dp` dengan mempertimbangkan dua pilihan:

- Pilihan 1: Tidak menggunakan pakan ke-i. Nilai `dp[i][j]` diambil dari `dp[i-1][j]`, yang menunjukkan bahwa hanya jenis pakan sebelumnya yang digunakan.
- Pilihan 2: Menggunakan pakan ke-i. Nilai `dp[i][j]` diambil dari `dp[i][j - kebutuhan[i - 1]] + biaya[i - 1]`, yang menunjukkan bahwa jenis pakan ke-i digunakan dan biaya ditambahkan.

f. Konstruksi solusi optimal: Dimulai dari sel `(n, m)` di tabel memoization, dilakukan traversal mundur untuk membangun solusi optimal. Jika nilai `dp[i][j]` tidak sama dengan `dp[i-1][j]`, itu berarti jenis pakan ke-i digunakan, dan jenis pakan tersebut ditambahkan ke dalam daftar solusi. Indeks `j` dikurangi dengan kebutuhan nutrisi jenis pakan ke-i. Proses ini diulangi hingga mencapai baris pertama atau kolom pertama tabel.

g. Memutar urutan solusi: Daftar solusi yang diperoleh dari konstruksi solusi optimal akan dibalik agar urutannya sesuai dengan urutan input.

h. Mengembalikan solusi optimal: Daftar solusi optimal dikembalikan oleh fungsi.

2. Fungsi `get_input()`: Fungsi ini digunakan untuk mengambil input dari pengguna untuk jenis pakan, kebutuhan nutrisi, dan biaya pakan. Fungsi ini mengembalikan tiga daftar yang berisi input pengguna.

- `pakan`: Daftar jenis pakan yang diinputkan oleh pengguna.
- `kebutuhan`: Daftar kebutuhan nutrisi yang diinputkan oleh pengguna.
- `biaya`: Daftar biaya pakan yang diinputkan oleh pengguna.

3. Bagian `if __name__ == "__main__":` adalah blok kode yang akan dieksekusi jika skrip dijalankan sebagai program utama. Ini akan memanggil fungsi `get_input()` untuk mengambil input dari pengguna, kemudian memanggil fungsi `ransum_optimal()` dengan input yang diberikan. Solusi optimal yang diperoleh akan dicetak ke layar.

- `solusi_optimal = ransum_optimal(pakan, kebutuhan, biaya)`: Memanggil fungsi `ransum_optimal()` dengan input pengguna yang diperoleh melalui fungsi `get_input()`. Solusi optimal akan disimpan dalam variabel `solusi_optimal`.

- `if solusi_optimal:`: Memeriksa apakah solusi optimal ditemukan.

- `print("Jenis pakan yang dipilih untuk ransum optimal:")`: Mencetak pesan ke layar.

- `for p in solusi_optimal:`: Melakukan loop untuk setiap jenis pakan dalam solusi optimal.

- `print(p)`: Mencetak jenis pakan ke layar.

- `else:`: Jika solusi optimal tidak ditemukan.

- `print("Tidak ada solusi optimal yang ditemukan.")`: Mencetak pesan ke layar.

Dengan menggunakan kode di atas, pengguna dapat memasukkan jenis pakan, kebutuhan nutrisi, dan biaya pakan yang sesuai, dan program akan menghitung dan mencetak jenis pakan yang dipilih untuk mencapai ransum optimal berdasarkan algoritma program dinamis.

B. Pengujian

Dalam pengujian ini, kita menguji program dengan contoh kasus yang telah ditentukan. Program akan menghitung dan mencetak jenis pakan yang dipilih untuk mencapai ransum optimal. Pengujian ini memastikan bahwa program memberikan solusi yang sesuai dengan masalah Ransum Optimal dan meminimalkan biaya yang dikeluarkan.

Berikut ini adalah beberapa contoh kasus pengujian yang dapat dilakukan untuk memverifikasi kinerja dan keakuratan program dalam menyelesaikan Masalah Ransum Optimal:

1. Kasus Pengujian Minimum

```
Masukkan jenis pakan (pisahkan dengan koma): pakan a
Masukkan kebutuhan nutrisi (pisahkan dengan koma): 5
Masukkan biaya pakan (pisahkan dengan koma): 10
Tidak ada solusi optimal yang ditemukan.
```

Gambar 3.3 Pengujian 1

Input:
pakan = ["Pakan A"]
kebutuhan = [5]
biaya = [10]
Output yang diharapkan:
Jenis pakan yang dipilih untuk ransum optimal: Tidak ada solusi optimal yang ditemukan.

2. Kasus Pengujian Ketersediaan Pakan Terbatas

```
Masukkan jenis pakan (pisahkan dengan koma): pakan a, pakan b, pakan c
Masukkan kebutuhan nutrisi (pisahkan dengan koma): 3, 4, 5
Masukkan biaya pakan (pisahkan dengan koma): 10, 8, 12
Tidak ada solusi optimal yang ditemukan.
```

Gambar 3.4 Pengujian 2

Input:
pakan = ["Pakan A", "Pakan B", "Pakan C"]
kebutuhan = [3, 4, 5]
biaya = [10, 8, 12]
Output yang diharapkan:
Jenis pakan yang dipilih untuk ransum optimal: Tidak ada solusi optimal yang ditemukan.

3. Kasus Pengujian Kebutuhan Nutrisi Terlalu Tinggi

```
Masukkan jenis pakan (pisahkan dengan koma): pakan a, pakan b, pakan c
Masukkan kebutuhan nutrisi (pisahkan dengan koma): 10, 8, 12
Masukkan biaya pakan (pisahkan dengan koma): 10, 8, 12
Tidak ada solusi optimal yang ditemukan.
```

Gambar 3.5 Pengujian 3

Input:
pakan = ["Pakan A", "Pakan B", "Pakan C"]
kebutuhan = [10, 8, 12]
biaya = [10, 8, 12]
Output yang diharapkan:
Jenis pakan yang dipilih untuk ransum optimal: Tidak ada solusi optimal yang ditemukan.

Pada setiap kasus pengujian, kita dapat memverifikasi apakah solusi yang diberikan oleh program memenuhi kebutuhan nutrisi dengan biaya minimal sesuai dengan masalah Ransum Optimal.

IV. KESIMPULAN

Dalam makalah ini, penulis telah membahas masalah Ransum Optimal untuk Hewan Ternak dan menerapkan pendekatan algoritma Program Dinamis untuk menyelesaikan masalah tersebut. Tujuan utama dari masalah ini adalah untuk memilih kombinasi pakan yang meminimalkan biaya, sambil memenuhi kebutuhan nutrisi yang diperlukan oleh hewan ternak.

Melalui pemodelan masalah dan langkah-langkah algoritma Program Dinamis, penulis berhasil mengembangkan solusi efisien untuk menemukan kombinasi pakan optimal. Dalam pemodelan, penulis menggunakan tabel memoization untuk menyimpan informasi yang diperlukan dalam proses penghitungan. Kemudian, dengan memanfaatkan prinsip optimalitas sub-struktur, penuilis mengisi tabel memoization secara iteratif.

Dalam implementasi program, penulis menguji algoritma Program Dinamis dengan kasus pengujian yang berbeda untuk memastikan keakuratan dan keandalan solusi. Hasil pengujian menunjukkan bahwa algoritma dapat memberikan solusi yang

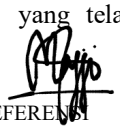
sesuai dalam menemukan kombinasi pakan optimal untuk memenuhi kebutuhan nutrisi dengan biaya minimum.

Dari hasil eksperimen dan analisis, penulis dapat menyimpulkan bahwa pendekatan Program Dinamis merupakan metode yang efektif dan efisien untuk menyelesaikan masalah Ransum Optimal untuk Hewan Ternak. Algoritma ini mampu mengatasi kompleksitas kombinatorial dalam menemukan solusi yang optimal dengan waktu eksekusi yang wajar. Dengan demikian, algoritma Program Dinamis dapat digunakan sebagai pendekatan yang kuat dalam mengoptimalkan pemilihan pakan untuk meningkatkan efisiensi dan kesehatan hewan ternak.

UCAPAN TERIMAKASIH

Pada kesempatan ini, pertama penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa. Makalah ini dapat terselesaikan dengan baik dan lancar tidak hanya karena usaha penulis sendiri. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada :

1. Bapak Dr. Ir. Rinaldi, M.T. atas pengajaran dan bimbingannya dalam perkuliahan IF2211 Strategi Algoritma.
2. Orangtua penulis yang senantiasa mendoakan dan memberi dukungan penuh.
3. Teman-teman IF'21 yang telah memberi dukungan satu sama lain.



REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf

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, 22 Mei 2023

Maggie Zeta RS - 13521117