Optimalisasi Pembagian Logistik Camping dengan Algoritma *Branch and Bound*

Reza Ahmad Syarif - 13523119[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]reza11781gmail.com, 13523119@std.stei.itb.ac.id

Abstrak— Makalah ini memformulasikan persoalan pembagian logistik pada kegiatan camping sebagai varian *Balanced Multi-Knapsack Problem* dan menyelesaikannya dengan algoritma *Branch and Bound*. Setiap peserta dipandang sebagai knapsack heterogen yang telah berisi beban pribadi, sedangkan item kelompok harus dialokasikan sedemikian rupa sehingga selisih beban maksimum–minimum (*spread*) antar peserta seminimal mungkin tanpa melampaui kapasitas ransel. Prosedur dimulai dengan heuristik greedy sebagai batas atas, diikuti penelusuran *Branch and Bound* dengan *lower-bound* relaksasi fraksional untuk memangkas ruang pencarian. Implementasi Python menunjukkan bahwa, untuk skala lapangan (≤ 5 peserta dan ≤ 10 item), algoritma mampu menghasilkan distribusi beban nyaris merata dengan spread ≤ 0,2 kg dan jumlah simpul eksplorasi terbatas. Pendekatan ini menyediakan kerangka terukur bagi panitia atau tim ekspedisi untuk merencanakan logistik secara adil, aman, dan efisien.

Kata Kunci— camping, logistik, multi-knapsack, branch and bound, pembagian beban, mendaki gunung.

I. PENDAHULUAN

Camping atau berkemah adalah kegiatan menginap di alam terbuka menggunakan tenda sebagai tempat tinggal sementara. Kegiatan camping umumnya dilakukan di lingkungan yang jauh dari perkotaan, seperti hutan, pantai, atau gunung. Kegiatan camping sering kali dikaitkan dengan kegiatan mendaki gunung, yaitu aktivitas menyelusuri jalur pendakian hingga mencapai puncak gunung. Kombinasi camping dan pendakian saat ini sedang menjadi tren di berbagai kalangan masyarakat, khususnya anak muda, sebagai bagian dari hobi, menantang diri, dan mencari udara segar untuk melepas penat dari kesibukan hiruk-pikuk kota.

Dalam pelaksanaannya, kegiatan camping dan mendaki gunung tidak dapat dipisahkan dari persiapan yang matang. Dalam mendaki gunung, persiapan fisik menjadi hal yang utama karena pendaki akan diuji dengan pendakian yang cukup lama dan jauh serta jalur yang memiliki sudut elevasi yang cukup curam. Selain persiapan fisik, pendaki juga perlu memperhatikan persiapan mental. Selama pendakian dan camping, pendaki akan diuji oleh banyak tantangan dan kejadian yang mungkin tidak terbayangkan sebelumnya, misalnya ketika cuaca buruk, berjalan malam hari, atau ketika ada kawan pendaki yang terkena cedera atau hipotermia. Oleh karena itu, pemahaman mendalam tentang medan pendakian, kondisi cuaca, dan durasi perjalanan menjadi bagian penting dalam persiapan kegiatan ini.

Selain persiapan mental dan fisik yang menjadi perhatian utama, persiapan logistik juga merupakan aspek yang sama pentingnya namun terkadang terlupakan. Persiapan logistik mencakup pemilihan, pengelolaan, serta distribusi barang-barang kebutuhan dasar seperti makanan, air, tenda, perlengkapan tidur, peralatan memasak, hingga perlengkapan medis. Ketersediaan dan pengelolaan logistik yang optimal sangat menentukan kelancaran dan kondisi fisik selama pendakian.

Terdapat satu kalimat populer di kalangan penggiat alam yaitu, "Logika tanpa Logistik tidak akan berjalan". Kalimat ini berarti bahwa sematang dan sebaiknya sebuah perencanaan pendakian, semuanya tidak akan berarti jika tidak disertai oleh perencanaan logistik yang efektif. Kalimat ini menekankan bahwa persiapan logistik juga merupakan aspek utama yang tidak kalah penting dari persiapan fisik dan mental saat melakukan kegiatan camping dan pendakian gunung. Kesiapan fisik, mental, dan juga logistik menjadi syarat minimal agar kegiatan camping dan pendakian gunung menjadi aman dan lancar. Tanpa persiapan yang tepat, pendaki bisa menghadapi berbagai risiko yang tidak diinginkan seperti cedera, kelelahan berlebihan, kelaparan, hipotermia, hingga ancaman keselamatan jiwa.

Dari latar belakang tersebut, makalah ini secara khusus bertujuan membahas strategi pembagian logistik secara optimal agar beban yang dibawa oleh setiap peserta dapat terbagi secara adil, efektif, dan tidak melebihi kapasitas yang dimiliki oleh masing-masing individu. Dalam menyelesaikan masalah ini, digunakan algoritma *Branch and Bound*, yaitu sebuah pendekatan matematis yang dikenal efektif dalam mencari solusi optimal untuk masalah kombinasi seperti pembagian logistik yang kompleks. Dengan implementasi algoritma ini, diharapkan dapat memberikan hasil nyata dalam membantu kegiatan camping menjadi lebih terencana, adil, aman, dan nyaman bagi semua peserta.

II. LANDASAN TEORI

A. Camping

Camping atau berkemah merupakan kegiatan rekreasi luar ruang yang dilakukan dengan tinggal sementara di alam terbuka menggunakan tenda. Kegiatan ini sering dilakukan di daerah yang jauh dari perkotaan seperti pegunungan, hutan, atau pantai. Camping mulai populer menjadi kegiatan rekreasi pada awal abad ke-20. Camping menjadi salah satu cara kebanyakan

masyarakat saat ini untuk melepas penat dari aktivitas perkotaan. Selain untuk rekreasi, camping juga digunakan sebagai sarana pelatihan karakter, penguatan kemandirian, dan kerja sama tim. Kegiatan camping umumnya disertai dengan kegiatan rekreasi luar ruangan lainnya seperti mendaki gunung, bersepeda, atau memancing.

Dalam konteks mendaki gunung, camping menjadi salah satu bagian penting dari perjalanan. Lokasi perkemahan umumnya telah ditetapkan oleh pengurus jalur pendakian, seperti Balai Besar Taman Nasional yang biasanya mengurus gunung-gunung berstatus Taman Nasional di Indonesia. Lokasi perkemahan juga dipilih pada titik-titik istirahat tertentu karena keterbatasan waktu dan tenaga untuk mencapai puncak dalam satu hari. Oleh karena itu, kegiatan pendakian membutuhkan perencanaan menyeluruh, baik dari segi rute, durasi, maupun kebutuhan logistik pendaki agar aktivitas berlangsung lancar, aman, dan nyaman.

B. Logistik

Logistik dalam kegiatan alam terbuka, khususnya dalam mendaki gunung, merujuk pada proses pengadaan, pengangkutan, penyimpanan, dan distribusi peralatan serta bahan makanan selama kegiatan berlangsung. Ketersediaan logistik yang lengkap dan tepat sasaran dapat meningkatkan keberhasilan serta keselamatan kegiatan. Distribusi logistik yang tidak merata dapat mengakibatkan beban berlebih salah satu anggota tertentu sehingga dapat menurunkan efisiensi gerak kelompok dan bahkan bisa menimbulkan risiko fisik. Oleh karena itu, prinsip logistik yang mengedepankan ketepatan jumlah, waktu, lokasi, dan kondisi sangat penting diterapkan meskipun dalam skala kecil seperti kegiatan camping. Pembagian beban yang merata dan sesuai kapasitas individu menjadi bagian penting dari manajemen logistik lapangan.

Menurut Buku Panduan Mendaki Gunung karya Ehwan Kurniawan, macam-macam barang logistik dapat dikelompokkan sebagai berikut.

1. Tempat berteduh: tenda dome, flysheet/terpal, pasak, dan tali.
2. Perlengkapan tidur: sleeping bag, matras, jaket, dan lilin.
3. Perlengkapan masak: kompor portabel, pemantik api, nesting, dan tempat air.
4. Pakaian: baju lapangan, celana lapangan, topi, sarung tangan, dan kacamata hitam.
5. Navigasi dan Komunikasi: peta topografi, kompas, GPS, peluit darurat, dan senter atau *headlamp*.
6. Keselamatan dan P3K: kotak P3K, *emergency blanket*, antiseptik, dan *sunblock*.

C. Algoritma Branch and Bound

Algoritma Branch and Bound (B&B) adalah teknik pencarian solusi optimal untuk persoalan optimisasi, baik minimisasi maupun maksimisasi, yang diselesaikan dengan membangun pohon ruang status secara dinamis. Setiap simpul pada pohon diberi sebuah nilai taksiran biaya $\hat{c}(i)$ yang merepresentasikan batas bawah (*lower bound*) terhadap nilai fungsi objektif terbaik yang masih mungkin dicapai lewat simpul tersebut. Selama proses, simpul hidup (belum dieksplor) disimpan dalam priority queue. Simpul yang diekspansi berikutnya adalah yang memiliki nilai $\hat{c}(i)$ terkecil (aturan *least-cost search*). Simpul yang tidak mungkin menghasilkan solusi yang lebih baik dari solusi saat ini akan dieliminasi (*bounding*) sehingga ruang pencarian bisa dipersempit secara signifikan.

Berikut adalah algoritma Branch and Bound secara umum:

1. Masukkan simpul akar ke antrean prioritas.
2. Selama antrean tidak kosong, pilih simpul dengan *cost* $\hat{c}(i)$ terkecil.
3. Jika simpul tersebut solusi, perbarui solusi terbaik.
4. Hitung bound setiap anak. Kemudian *prune* anak yang *bound*-nya tidak bisa mengalahkan solusi terbaik atau melanggar batasan masalah.
5. Masukkan anak-anak tersisa ke antrean dan ulangi sampai antrean habis.

Pendekatan Branch and Bound sangat berguna dalam menyelesaikan masalah seperti penjadwalan, *Travelling Salesman Problem*, dan pembagian beban, yang memiliki jumlah kombinasi solusi sangat besar. Algoritma B&B dapat digunakan untuk memecahkan masalah pembagian logistik ke dalam beberapa knapsack dengan batasan kapasitas, dengan tujuan meminimalkan perbedaan beban antar peserta. Keunggulan algoritma ini adalah kemampuannya dalam menjamin solusi optimal selama fungsi pembatas (*bounding function*) dirancang secara efisien.

D. Multi-Knapsack Problem

Multi-Knapsack Problem (MKP) merupakan pengembangan dari masalah klasik 0/1 Knapsack yang melibatkan lebih dari satu knapsack dengan kapasitas tertentu. Tujuan dari MKP adalah untuk mengalokasikan sejumlah item ke dalam beberapa knapsack tanpa melebihi kapasitas masing-masing dengan sasaran tertentu seperti memaksimalkan nilai total atau mendistribusikan beban secara seimbang. MKP termasuk dalam kelas masalah NP-Hard sehingga penyelesaiannya tidak dapat dilakukan secara langsung untuk skala besar tanpa pendekatan algoritmik khusus.

Dalam konteks pembagian logistik camping, setiap peserta dipandang sebagai satu knapsack dengan kapasitas terbatas dan item-item logistik kelompok perlu dibagi secara efisien ke dalam knapsack-knapsack tersebut. Penyelesaian MKP dapat dilakukan dengan berbagai pendekatan, termasuk algoritma *exact* seperti Dynamic Programming dan Branch and Bound, maupun pendekatan heuristik dan metaheuristik.

E. Balanced Load Distributuion

Balanced Load Distribution adalah konsep distribusi beban secara merata ke sejumlah pembawa dengan tujuan meminimalkan ketimpangan. Dalam konteks pendakian dan camping, setiap individu memiliki keterbatasan fisik dan kapasitas ransel yang berbeda. Oleh karena itu, distribusi beban harus dirancang sedemikian rupa agar tidak ada satu peserta pun yang membawa beban berlebih, sementara yang lain terlalu ringan. Ketimpangan seperti ini tidak hanya berpengaruh terhadap stamina individu, tetapi juga pada efektivitas kerja sama tim.

Balanced Load Distribution banyak dipelajari dalam konteks logistik militer, industri distribusi barang, bahkan dalam penjadwalan komputasi paralel. Salah satu cara umum untuk

mengukur keseimbangan beban adalah dengan meminimalkan *spread*, yaitu selisih antara beban maksimum dan minimum antar pembawa. Dalam konteks logistik camping, strategi algoritmik diarahkan untuk meminimalkan *spread* tersebut dengan tetap memperhatikan batasan kapasitas, menjadikan proses pembagian logistik tidak hanya optimal, tetapi juga adil dan dapat diterima oleh seluruh anggota tim.

III. Analisis Dan Implementasi

A. Analisis Permasalahan

Masalah pembagian logistik dalam kegiatan camping dapat dianalisis dan dimodelkan sebagai bentuk khusus dari Multi-Knapsack Problem (MKP). Dalam formulasi klasik MKP, terdapat sejumlah item yang harus dimasukkan ke dalam beberapa knapsack dengan kapasitas tertentu, dengan tujuan untuk memaksimalkan nilai total dari item yang dimasukkan. Namun, dalam konteks kegiatan camping, fokus permasalahan bukanlah pada nilai item, melainkan pada keseimbangan beban antar pembawa agar distribusi logistik adil dan tidak memberatkan pihak tertentu. Untuk menjembatani konsep ini, dilakukan penyesuaian formulasi MKP menjadi varian *balanced multi-knapsack*, yang memiliki tujuan optimisasi untuk meminimalkan selisih beban antar knapsack (*spread*) dengan tetap mematuhi batas kapasitas masing-masing.

1. Peserta
   Dalam konteks ini, setiap peserta camping dimodelkan sebagai sebuah knapsack dengan kapasitas tertentu. Masing-masing peserta *i* memiliki:
   - Beban awal pribadi (P_i), yaitu beban yang sudah dibawa sejak awal, seperti pakaian, air minum, dan peralatan tidur.
   - Kapasitas maksimum (C_i), yaitu batas maksimal beban total yang dapat mereka bawa berdasarkan kekuatan fisik dan ukuran carrier.

   Sisa kapasitas aktual yang dapat digunakan untuk membawa logistik kelompok dihitung sebagai:
   $$R_i = C_i - P_i$$
   Knapsack dianggap valid jika $R_i > 0$. Peserta dengan $R_i \leq 0$ tidak dapat dibebani logistik tambahan.
2. Item Logistik Kelompok
   Setiap barang logistik kelompok diasumsikan sebagai sebuah item tunggal tak terbagi, sesuai dengan asumsi standar pada MKP. Item tersebut memiliki atribut utama berupa:
   - Bobot (w_j), yaitu berat dalam satuan kilogram.

   Barang-barang ini merupakan kebutuhan bersama, seperti tenda, alat masak, makanan kelompok, atau kotak P3K. Setiap item harus diberikan persis ke satu peserta, tanpa bisa dibagi ke lebih dari satu orang.
3. Batasan Permasalahan
   Permasalahan yang dibahas memiliki dua jenis batasan utama, yaitu:
   - Batasan Kapasitas Individu (*knapsack constraint*)
     Setiap peserta tidak boleh membawa beban melebihi kapasitas ranselnya:
     $$P_i + \sum_{\{j=1\}}^{m} x_{ij} \cdot w_j \leq C_i, \forall i$$
   - Batasan distribusi unik item (*assignment contraint*)
     Setiap item hanya boleh diberikan ke satu peserta:
     $$\sum_{\{i=1\}}^{n} x_{ij} = 1, \forall j$$
     Di sini, x_{ij} adalah variabel biner yang menyatakan apakah item j diberikan kepada peserta i atau tidak.
4. Fungsi Tujuan
   Tidak seperti MKP klasik yang berfokus pada maksimisasi nilai total, tujuan dari permasalahan ini adalah untuk mencapai distribusi beban yang seimbang antar peserta. Oleh karena itu, fungsi objektifnya adalah meminimalkan *spread* beban:
   $$Spread = \max_i(P_i + \sum_j x_{ij} \cdot w_j) - \min_i(P_i + \sum_j x_{ij} \cdot w_j)$$
   Nilai *spread* yang kecil menunjukkan bahwa beban peserta relatif merata. Pendekatan ini tergolong *balanced* MKP, yang merupakan turunan dari MKP dengan tujuan pemerataan beban alih-alih total keuntungan.

Untuk menyelesaikan permasalahan *balanced multi-knapsack* ini, digunakan algoritma *Branch and Bound* (B&B). Berikut adalah algoritma *Multi-Knapsack Problem* dengen pendekatan *Branch and Bound*.

1. Pra-proses & Upper Bound
   Urutkan item menurun bobot (*largest-first*) untuk mempercepat deteksi *infeasible*.. Kemudian, jalankan heuristik *greedy balancing* guna menghasilkan upper bound (UB) awal terhadap spread.
2. Struktur Pohon Keputusan
   Tiap node mewakili status setelah item ke-*k* ditempatkan. State menyimpan *mask* item yang telah didistribusikan, *vector* beban L_i, serta maksimal & minimal *load* saat itu.
3. Branching
   Untuk item ke-*k,* cabangkan ke setiap peserta yang masih memiliki kapasitas R_i cukup.
4. Bounding
   Lower Bound (LB) dihitug sebagai:
   LB = max(*current_max*, $\min_i L_i + \left\lceil \frac{berat\ sisa}{n} \right\rceil$) - $\min_i L_i$
   Jika LB ≥ UB, maka cabang dipangkas (*pruning*).
5. Pencatatan Solusi Terbaik
   Ketika semua item sudah teralokasi, hitung *spread*. Perbarui UB dan simpan *assigment* jika *spread* lebih kecil.

Agar permasalahan pembagian logistik dengan menggunakan *Branch and Bound* ini lebih tergambar, berikut adalah contoh kasusnya.

Peserta	**Beban Pribadi P_i (kg)**	**Kapasitas C_i (kg)**

Aryo	5.0	15.0
Bama	6.0	17.0
Cihuy	4.0	14.0

Item	Berat w_i (kg)
Tenda	3.2
Kompor	1.5
Gas	1.2
Sosis	2.0
Nugget	2.5
P3K	0.8
Flysheet	1.0

Langkah Eksekusi:

1. Heuristik Greedy (Upper Bound): Urutkan seluruh item dari berat terbesar ke terkecil. Untuk setiap item, pilih peserta yang memiliki beban terkecil. Jika terjadi beban yang sama, pemilihan dipecahkan secara alfabetis.

Urutan	Item (kg)	Peserta Terpilih	Beban
0	-	-	Aryo: 5,0 Bama: 6,0 Cihuy: 4,0
1	Tenda 3,2	Cihuy	Aryo: 5,0 Bama: 6,0 Cihuy: 7,2
2	Nugget 2,5	Aryo	Aryo: 7,5 Bama: 6,0 Cihuy: 7,2
3	Sosis 2,0	Bama	Aryo: 7,5 Bama: 8,0 Cihuy: 7,2
4	Kompor 1,5	Cihuy	Aryo: 7,5 Bama: 8,0 Cihuy: 8,7
5	Gas 1,2	Aryo	Aryo: 8,7 Bama: 8,0 Cihuy: 8,7
6	Flysheet 1,0	Bama	Aryo: 8,7 Bama: 9,0 Cihuy: 8,7
7	P3K 0,8	Aryo	Aryo: 9,5 Bama: 9,0 Cihuy: 8,7

   - Spread awal = 9,5 − 8,7 = 0,8 kg.
   - Nilai 0,8 kg digunakan sebagai batas atas pertama ke algoritma Branch and Bound.
2. Inisialisasi Branch & Bound: Buat node akar dengan k=0, loads = (5,0; 6,0; 4,0), dan LB = 0. Masukkan ke *priority queue* sebagai (LB, −k, loads, assignment).
3. Proses Iteratif:
   - Pop node dengan LB terkecil.
   - Jika k < m, hitung bound optimistik via *fractional relaxation* atas sisa item. Jika LB ≤ UB, branch untuk tiap peserta yang masih muat, buat node baru dengan k+1, loads diperbarui, dan hitung LB-nya. Jika LB > UB, prune node.
   - Jika k == m, hitung *spread* riil. Jika < UB, perbarui UB dan simpan *assignment* sebagai solusi terbaik
4. Perketat Upper Bound:
   - Saat ditemukan node lengkap dengan *spread* riil 0,20 kg, set UB = 0,20.
   - emua node tersisa dengan LB ≥ 0,20 segera dipangkas.
5. Terminasi
   Ketika *priority queue* kosong, assignment terbaik adalah solusi akhir. Hasil akhir yang didapatkan sebagai berikut.

Peserta	Item kelompok	Beban total
Aryo	Tenda, P3K	5,0 + 3,2 + 0,8 = 9,0 kg
Bama	Sosis, Flysheet	6,0 + 2,0 + 1,0 = 9,0 kg
Cihuy	Nugget, Kompor, Gas	4,0 + 2,5 + 1,5 + 1,2 = 9,2 kg
Spread akhir		**0,20 kg**

Hasilnya, *spread* (beban maksimal − beban minimal) = 9,2 kg – 9,0 kg = 0,2 kg setelah optimal. Beban Aryo seberat 9,0 kg, Budi seberat 9,0 kg, dan Cici seberat 9,2 kg. Seluruh kapasitas terpenuhi tanpa melampaui batas dan beban hampir merata.

B. Implementasi

Implementasi ini bertujuan untuk memetakan permasalahan pembagian beban logistik camping ke dalam kerangka *balanced multi-knapsack* menggunakan algoritma *Branch and Bound*. Program ini dirancang untuk membantu pengguna mendapatkan penugasan distribusi item yang optimal berdasarkan parameter beban pribadi dan kapasitas *carrier*, serta menghitung nilai *spread* sebagai metrik keadilan distribusi.

Dalam implementasi ini, terdapat tiga komponen utama dengan penjelasan sebagai berikut..

1. Definisi Tipe Data
   Tipe data *Camper* mewakili setiap peserta dengan atribut *name*, personal (beban awal pribadi P_i), dan *capacity* (kapasitas maksimum *carrier* C_i). Tipe data Item mewakili setiap barang logistik kelompok, dengan atribut *code* dan *weight* (w_j).

```python
# ---------- Tipe Data ----------
@dataclass
class Camper:
    name: str          # nama peserta
    personal: float    # beban awal pribadi (P_i)
    capacity: float    # kapasitas maksimum carrier (C_i)

@dataclass
class Item:
    code: str          # kode item
    weight: float      # berat item (w_j)
```

2. Greedy Balancer
   Fungsi greedy_balance(campers, items, reverse) melakukan pembagian item satu per satu ke peserta dengan beban terkecil yang masih muat. Parameter

reverse mengubah urutan (terbesar-terkecil dan juga sebaliknya) untuk mendapatkan dua solusi dengan *heuristik* berbeda. Outputnya adalah nilai *spread* awal dan pembagian indeks item per peserta yang akan digunakan sebagai batas atas atau upper bound (UB) awal pada algoritma *Branch & Bound*.

```python
# ---------- Greedy Balancer sebagai penghitung batas atas ----------
def greedy_balance(campers: List[Camper], items: List[Item],
                   reverse: bool = False) -> Tuple[float, List[List[int]]]:
    """
    Pembagian secara Greedy untuk mendapatkan batas atas.
    Jika reverse=False, maka urutan terbesar-terkecil.
    Jika reverse=True, maka terkecil-terbesar.
    Mengembalikan (spread, pembagian) dengan daftar pembagian indeks item asli.
    """
    n, m = len(campers), len(items)
    loads = [c.personal for c in campers]
    caps  = [c.capacity for c in campers]
    assignment = [[] for _ in campers]

    # Mensortir item berdasarkan berat sesuai dengan flag
    order = sorted(range(m), key=lambda i: items[i].weight, reverse=not reverse)
    for idx in order:
        w = items[idx].weight
        # Mencari kandidat peserta yang dapat membawa barang idx
        candidates = [(loads[i], i) for i in range(n) if loads[i] + w <=
caps[i]]if not candidates:
            return math.inf, []
        # Mengambil peserta yang memiliki beban paling minimum saat ini
        _, chosen = min(candidates)
        loads[chosen] += w
        assignment[chosen].append(idx)
    spread = max(loads) - min(loads)
    return spread, assignment
```

3. Branch & Bound

Fungsi utama branch_and_bound(campers, items) untuk menyelesaikan masalah *balanced multi-knapsack*. Sebelum proses B&B dilakukan *pre*-check terlebih dahulu untuk memastikan tidak ada item yang lebih berat daripada kapasitas sisa peserta mana pun sehingga akan langsung menolak instansi tidak *feasible*.

```python
# ---------- Branch & Bound ----------
def branch_and_bound(campers: List[Camper], items: List[Item],
                     verbose: bool = True) -> Tuple[dict, float]:
    n, m = len(campers), len(items)
    base = [c.personal for c in campers]
    resid = [c.capacity - c.personal for c in campers]

    # feasibility check
    if any(it.weight > max(resid) for it in items):
        raise ValueError("Ada item lebih berat dari sisa terbesar.")

    # upper-bound awal
    best_spread, _ = min(
        (greedy_balance(campers, items, rev) for rev in (False, True)),
        key=lambda x: x[0]
    )

    order = sorted(range(m), key=lambda i: -items[i].weight)
    w_ord = [items[i].weight for i in order]

    heap: List[Tuple[float, int, Tuple[float, ...],
Tuple[Tuple[int,...],...]]] = []
    heapq.heappush(heap, (0.0, 0, tuple(base), tuple(() for _ in campers

    best_assign = None
    step = 0

    while heap:
        lb, neg_k, loads, assign = heapq.heappop(heap)
        k = -neg_k
        step += 1

        if verbose:
            print(f"[Step {step:3d}] k={k}  lb={lb:.2f}  loads=
{tuple(round(x,1) for x in loads)}")

        if lb > best_spread:
            continue
        if k == m:
            spread = max(loads) - min(loads)
            if spread < best_spread:
                best_spread, best_assign = spread, assign
            continue
```

```python
        # bound optimistik
        rem_w   = sum(w_ord[k:])
        avg_tot = (sum(loads) + rem_w) / n
        cur_max, cur_min = max(loads), min(loads)
        lb_node = max(cur_max, avg_tot) - min(cur_min + rem_w, avg_tot)
        if lb_node > best_spread:
            continue

        # branching
        idx_item = order[k]
        w = items[idx_item].weight
        for i in range(n):
            if loads[i] + w - base[i] > resid[i]:
                continue
            new_loads = list(loads); new_loads[i] += w
            new_assign = list(assign); new_assign[i] = assign[i] +
(idx_item,)
            heapq.heappush(heap,
                (lb_node, -(k+1), tuple(new_loads), tuple(new_assign)))

    if best_assign is None:
        raise ValueError("Instansi tidak feasible.")

    if verbose:
        print(f"\nTotal node diekspansi: {step}")

    # build hasil
    result = {
        campers[i].name: [items[j].code for j in best_assign[i]]
        for i in range(n)
    }
    return result, best_spread
```

Berikut adalah alur logika program dengan contoh kasus.

1. Inisialisasi Data

   Load daftar peserta dan daftar barang.

```python
campers = [
    Camper("Aryo", 5.0, 15.0),
    Camper("Bama", 6.0, 17.0),
    Camper("Cihuy", 4.0, 14.0)
]
items = [
    Item("Tenda", 3.2),
    Item("Kompor", 1.5),
    Item("Gas", 1.2),
    Item("Sosis", 2.0),
    Item("Nugget", 2.5),
    Item("P3K", 0.8),
    Item("Flysheet", 1.0)
]

sol, spread = branch_and_bound(campers, items)
```

2. Validasi Awal

   Cek setiap item apabila ada yang terlalu berat untuk semua peserta, program langsung error.

```python
max_res = max(residual_caps)
for it in items:
    if it.weight > max_res:
        raise ValueError(f"Item '{it.code}' (w={it.weight}) terlalu berat
untuk semua camper.")
```

3. Menentukan Upper Bound

   Fungsi greedy_balance dijalankan dua kali (urutan terbesar-ke-terkecil & sebaliknya) untuk mendapatkan best_spread terkecil sebagai batas atas awal untuk proses pruning.

```python
    ub1, _ = greedy_balance(campers, items, reverse=False)
    ub2, _ = greedy_balance(campers, items, reverse=True)
    best_spread = min(ub1, ub2)
```

4. Mempersiapkan Struktur *Branch & Bound*

   Indeks item diurutkan berdasarkan berat menurun. Kemudian, dibuat priority queue berisi node root dengan beban awal pribadi dan assignment kosong.

```python
    order = sorted(range(m), key=lambda i: -items[i].weight)
    w_ordered = [items[i].weight for i in order]

    init_assign = tuple(() for _ in campers)
    heap = [(0.0, 0, tuple(base_loads), init_assign)]
    best_assign = None
```

5. Proses Branch & Bound
   - Ambil node terbaik yaitu node dengan lower bound terkecil dari heap.
   - Lakukan pruning apabila bound > UB.
   - Jika semua item sudah dialokasikan, hitung spread dan perbarui solusi terbaik.
   - Jika belum lengkap, hitung bound optimistik, lalu untuk setiap peserta yang masih muat, buat node anak dengan beban terbaru dan assignment yang diperbarui.
   - Ulangi hingga heap kosong atau semua kemungkinan yang berguna sudah diproses.

```
while heap:
    lb, neg_k, loads, assign = heapq.heappop(heap)
    k = -neg_k

    if lb > best_spread:
        continue

    if k == m:
        spread = max(loads) - min(loads)
        if spread < best_spread:
            best_spread, best_assign = spread, assign
        continue

    rem_w = sum(w_ordered[k:])
    sum_now = sum(loads)
    avg_total = (sum_now + rem_w) / n
    cur_max, cur_min = max(loads), min(loads)
    opt_max = max(cur_max, avg_total)
    opt_min = min(cur_min + rem_w, avg_total)
    lb_node = opt_max - opt_min

    if lb_node > best_spread:
        continue

    idx_item = order[k]
    w = items[idx_item].weight
    for i in range(n):
        if loads[i] + w - base_loads[i] > residual_caps[i]:
            continue
        new_loads = list(loads)
        new_loads[i] += w
        new_assign = list(assign)
        new_assign[i] = assign[i] + (idx_item,)
        heapq.heappush(
            heap,
            (lb_node, -(k+1), tuple(new_loads), tuple(new_assign))
        )
```

6. Susun Hasil
   Jika ada solusi (*best_assign*), konversi indeks-item menjadi kode barang (Item.code). Kembalian berupa struktur dictionary berisi nama peserta, daftar kode item, dan nilai best_spread.

```
result = {
    campers[i].name: [items[j].code for j in best_assign[i]]
    for i in range(n)
}
return result, best_spread
```

7. Cetak Hasil
   Hasil solusi ditampilkan per peserta dan disertai total beban serta nilai spread akhir.

```
=== HASIL BRANCH & BOUND ===
Aryo : ['Tenda', 'P3K'] | total ≈ 9.0 kg
Bama : ['Sosis', 'Flysheet'] | total ≈ 9.0 kg
Cihuy: ['Nugget', 'Kompor', 'Gas'] | total ≈ 9.2 kg
Spread beban akhir: 0.20 kg
```

IV. KESIMPULAN

Algoritma *Branch and Bound* terbukti efektif memecahkan masalah pembagian logistik camping dengan pendekatan *balanced multi-knapsack*. Dengan memodelkan kapasitas ransel, beban pribadi, dan bobot tiap barang, program mampu menghasilkan distribusi beban yang adil dengan nilai spread yang kecil dalam waktu singkat. Solusi heuristik greedy sebagai batas atas dan bound relaksasi fraksional membantu memangkas ruang pencarian tanpa kehilangan optimalitas. Hasil demonstrasi menunjukkan setiap peserta membawa beban mendekati seimbang sehingga dapat meminimalkan risiko kelelahan di lapangan. Harapannya, algoritma dapat diperluas dengan dimensi volume atau preferensi item untuk skenario ekspedisi yang lebih kompleks.

REFERENSI

[1] R. Munir, Algoritma branch and bound (Bagian 1). Bandung: Informatika, 2025. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/ pada tanggal 19 Juni 2025 pukul 20.00.

[2] R. Munir, Algoritma branch and bound (Bagian 4). Bandung: Informatika, 2025. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/ pada tanggal 19 Juni 2025 pukul 20.00.

[3] G. GaluhRegaPramadya, "Buku Panduan Mendaki Gunung.pdf," SlideShare, 15 Desember 2022. [Online]. Diakses dari https://www.slideshare.net/slideshow/buku-panduan-mendaki-gunungpdf/254904786 pada tanggal 20 Juni 2025 pukul 19.00.

[4] H. Kellerer, U. Pferschy, dan D. Pisinger, "Introduction to NP-Completeness of Knapsack Problems," [Online]. Diakses dari https://link.springer.com/chapter/10.1007/978-3-540-24777-7_16 pada tanggal 20 Juni 2025 pukul 17.00.

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, 24 Juni 2025

Reza Ahmad Syarif / 13523119