Problem Ranca (Knapsack Problem)

Nikola Pantić IN 40/2020

January 13, 2024

Sadržaj

Uvod u Knapsack Problem

Knapsack problem je klasičan problem optimizacije sa širokom primenom u računarskoj nauci i oblasti operacionih istraživanja.

Osnovna Verzija Problema

Imate ranac određenog kapaciteta i skup predmeta, pri čemu svaki predmet ima određenu težinu i vrednost. Cilj je odabrati podskup predmeta tako da ukupna težina ne premašuje kapacitet ranca, a ukupna vrednost podskupa je maksimalna. Ovaj problem često susrećemo u kontekstu maksimizacije vrednosti pri ograničenjima težine.

Različite Varijante Knapsack Problema

Postoje različite varijante Knapsack problema, koje zahtevaju prilagođene pristupe rešavanju:

- **0/1 Knapsack Problem:** Predmeti se mogu ili ne mogu uzimati u celi u ranac (ne možete deliti predmete).
- **Fractional (ili Continuous) Knapsack Problem:** Predmeti se mogu deliti i delimično smestiti u ranac, proporcionalno njihovoj vrednosti.
- **Unbounded Knapsack Problem:** Predmeti se mogu uzimati u neograničenim količinama.

Svaka varijanta ima svoje specifičnosti i zahteva različite strategije rešavanja. Ova fleksibilnost čini Knapsack problem izuzetno korisnim u raznim situacijama optimizacije.

Napomena: Mi ćemo se fokusirati na rešavanje 0/1 Knapsack problema.

Greedy Algoritam

Uvod

Greedy algoritam za rešavanje knapsack problema predstavlja jednostavan pristup izboru predmeta na osnovu njihovog odnosa vrednosti i težine, poznatog kao profitabilnost. Ovaj algoritam preferira predmete sa najvećim profitabilnostima i nastoji ih smestiti u ranac dokle god ima dovoljno slobodnog prostora.

Razrada

1. **Sortiranje predmeta:** Predmeti se sortiraju u opadajućem redosledu odnosa vrednosti i težine, tj. njihove profitabilnosti. Ovaj korak omogućava algoritmu da prvo razmatra najprofitabilnije predmete.

2. **Iteracija kroz sortirane predmete:** Algoritam iterira kroz sortirane predmete i razmatra ih redom. Ovaj korak omogućava algoritmu da donosi odluke na osnovu trenutnog stanja ranca i profitabilnosti predmeta.

3. **Dodavanje predmeta u ranac:** Predmeti se dodaju u ranac sve dok ima dovoljno slobodnog mesta. Ovaj korak se ponavlja dok se ne popuni sav dostupan prostor u rancu ili dok ne ostanu predmeti koji se mogu dodati.

Tumačenje rezultata

Algoritam generiše rešenje koje može biti optimalno u određenim slučajevima, ali nije garantovano optimalno u svim situacijama. Greedy algoritmi često ne pružaju globalno optimalna rešenja, jer se fokusiraju na trenutno najbolje rešenje, a ne na dugoročnu optimizaciju.

Zaključak

Greedy algoritam za knapsack problem ističe se po brzini i jednostavnosti. Međutim, važno je napomenuti da ne garantuje uvek optimalno rešenje.
Iako ovaj algoritam može pružiti dobra rešenja za određene instance problema, za kompleksnije instance može biti potrebno koristiti naprednije tehnike, kao što su dinamičko programiranje ili branch-and-bound metode.

Branch and Bound Algoritam

Uvod

Branch and Bound algoritam predstavlja efikasan pristup rešavanju knapsack problema. Ovaj algoritam primenjuje strategiju pretraživanja prostora stanja kako bi pronašao optimalno rešenje. Ključni elementi ovog pristupa uključuju heuristiku za procenu gornje granice vrednosti rešenja, poznatu kao "bound funkcija", i primenu strategije grananja i ograničenja.

Razrada

1. **Sortiranje predmeta:** Predmeti se sortiraju u opadajućem redosledu odnosa vrednosti i težine, kako bi se poboljšala efikasnost algoritma. Ovaj korak omogućava algoritmu da prvo razmatra predmete sa najvećom profitabilnošću.

2. **Heuristika za procenu gornje granice:** Koristi se heuristika za procenu gornje granice vrednosti rešenja, poznata kao "bound funkcija". Ova funkcija pomaže algoritmu da odredi vrednost koju može očekivati od potencijalnih rešenja na određenom nivou pretrage. Ova procena se koristi za odlučivanje da li je vredno dalje istraživati određenu granu prostora stanja.

3. **Strategija grananja i ograničenja:** Primenjuje se strategija grananja i ograničenja kako bi se pretraživao prostor stanja. Algoritam odlučuje koje grane prostora stanja treba dalje istražiti, uzimajući u obzir heurističku procenu gornje granice. Ova strategija omogućava algoritmu da efikasno pretražuje prostor stanja, istražujući samo one grane koje imaju potencijal da pruže optimalno rešenje.

Tumačenje rezultata

Branch and Bound algoritam garantuje pronalaženje optimalnog rešenja, što znači da će pronaći najveću moguću vrednost za knapsack problem. Ipak, važno je napomenuti da ovakav pristup može zahtevati znatno više resursa u poređenju sa jednostavnijim algoritmima, poput Greedy pristupa. Ovo je zbog činjenice da Branch and Bound algoritam mora da pretraži veliki prostor stanja kako bi garantovao optimalnost.

Zaključak

Branch and Bound pristup se ističe po efikasnosti u rešavanju knapsack problema, ali se može pokazati resursno zahtevnim u određenim situacijama. Njegova garancija optimalnog rešenja čini ga odličnim izborom u situacijama gde je postizanje maksimalne vrednosti prioritet. Međutim, u situacijama gde su resursi ograničeni, može biti potrebno razmotriti alternativne pristupe, kao što su Greedy algoritam ili dinamičko programiranje.

Genetski Algoritam

Uvod

Genetski algoritam predstavlja pristup rešavanju optimizacionih problema inspirisan principima prirodne evolucije i genetike. U kontekstu Knapsack problema, genetski algoritam pokušava pronaći optimalni podskup predmeta koji se smešta u ranac, maksimizirajući ukupnu vrednost predmeta a da pritom ne premašuje ukupnu dozvoljenu težinu.

Razrada

1. **Inicijalizacija populacije:** Kreiranje početne populacije jedinki sa nasumičnim genomima (0 ili 1).
2. **Genetski operatori:**
   - **Ukrštanje (Crossover):** Primena operatora ukrštanja na parove roditelja kako bi se generisali potomci. Ukrštanje se obično vrši na tački preseka (single-point crossover) ili na više tačaka preseka (multi-point crossover).
   - **Mutacija:** Promena genetskog materijala pojedinih jedinki sa određenom verovatnoćom. Mutacija se obično vrši promenom vrednosti pojedinačnih gena.
3. **Evaluacija fitnesa:** Procena prilagođenosti svake jedinke u populaciji na osnovu zadatih kriterijuma. U ovom slučaju, fitnes funkcija se obično računa kao ukupna vrednost predmeta u rancu, uzimajući u obzir da ukupna težina ne premašuje dozvoljenu težinu.
4. **Evolucija populacije:** Selekcija roditelja, primena ukrštanja i mutacije, te formiranje naredne generacije. Selekcija se obično vrši na osnovu fitnesa, gde jedinke sa većim fitnesom imaju veću šansu da budu izabrane za roditelje.

Tumačenje rezultata

Iako genetski algoritam ima potencijal da pronađe globalno optimalno rešenje, to nije garantovano. Efikasnost algoritma zavisi od parametara kao što su veličina populacije, verovatnoća mutacije, i drugi faktori.

Zaključak

Genetski algoritam pruža fleksibilan i robustan pristup rešavanju Knapsack problema. Podešavanje parametara, kao i pravilna implementacija genetskih operatora, ključni su za postizanje dobrih rezultata. Ovaj algoritam je posebno koristan u situacijama gde je prostor rešenja kompleksan i gde se traži brza konvergencija ka optimalnom rešenju