به نام یگانه معبود بخشنده مهربان

طراحی الگوریتم ها

Design and Analysis of Algorithms

گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه اصفهان

ترم دوم سال تحصیلی ۹۰–۹۱

ارائه دهنده : پیمان ادیبی

روش شاخه و حد

Branch-and-Bound

مفاهیم

- این روش مانند روش عقبگرد برای جستجوی کارآمد درخت فضای حالت بکار میرود، با این تفاوت که :
  - تنها در مسایل بهینه سازی بکار میرود.
  - جستجو محدود به روش عمق اول نیست

- مانند روش عقبگرد، روش شاخه و حد نیز در بدترین حالت معمولاً پیچیدگی نمایی یا بدتر دارد، اما در عمل برای نمونه های بزرگ میتواند بسیار کارآمد باشد.

جستجوی بهترین اول با هرس کردن شاخه و حد

- یک نسخه اصلاح شده از **BFS** در مسایل بهینه سازی است.

```
void best_first_branch_and_bound ( state_space_tree T,
                                   number& best
{
  priority_queue_of_node PQ;
  node u, v;

  initialize (PQ);                          // Initialize PQ to be empty.
  v = root of T;
  best = value(v);
  insert (PQ, v);
  while (! empty (PQ)) {                    // Remove node with best
     remove (PQ, v);                        // bound.

   if (bound (v) is better than best)          // Check if node is still
         for (each child u of v) {            // promising.
                if (value(u) is better than best)
                       (best = value (u);
                if (bound (u) is better than best)
                       insert (PQ, u);
         }
   }
}
```

جستجوی بهترین اول با هرس کردن شاخه و حد

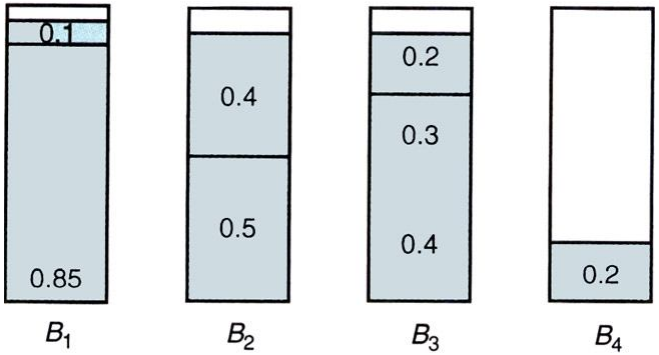
- در این الگوریتم در هر مرحله از میان گره های امیدبخش بسط نیافته همواره آن که بهترین **bound** را دارد برای بسط در سطح بعد انتخاب میشود.
- مثال عددی (کوله پشتی صفر و یک): ...

الگوریتمهای تقریب

Approximation Algorithms

مسأله بسته بندی (Bin-Packing)

- میخواهیم تعداد n شیء با حجم های s_1، s_2، ...، و s_n ($0<s_i\leq 1$) را در جعبه هایی با گنجایش 1 بسته بندی کنیم به طوری که حداقل تعداد جعبه ممکن بکار رود.
- مسأله NP مشکل است.
- الگوریتم ساده زیر (اولین جای دهی غیر افزایشی) یک پاسخ تقریبی برای این مسأله در زمان چندجمله‌ای بدست می‌آورد:
  - ابتدا اشیاء را به ترتیب غیر افزایشی حجم شان مرتب میکنیم.
  - اولین شیء را در اولین جعبه (موجود) که جا بشود قرار میدهیم.
  - اگر چنین جعبه ای موجود نبود، یک جعبه جدید بدین منظور برمیداریم.
- مثال زیر نشان میدهد که این الگوریتم تقریب لزوماً بهینه عمل نمیکند: $s_1=0.85, s_2=0.5, s_3=0.4, s_4=0.4, s_5=0.3, s_6=0.2, s_7=0.2$
- نشان داده میشود که یک حد بالای مناسب برای خطای تقریب این الگوریتم وجود دارد.



مسأله بسته بندی (Bin-Packing)

- الگوریتم اولین جای دهی غیرافزایشی (*nonincreasing first fit*):

```
sort the items in nonincreasing order;
while (there are still unpacked items ){
        get next item;
        while (the item is not packed and there are more started bins){
                get next bin;
                if(the item fits in the bin)
                        pack it in the bin
        }
        if (the item is not packed){
                start a new bin;
                place the item in the new bin;
        }
}
```

- **لم 9.1**: اگر opt تعداد جعبه بهینه برای یک نمونه مسأله بسته بندی باشد، آنگاه حجم هر شیء که توسط الگوریتم فوق در یک جعبه اضافی (یعنی جعبه ای با اندیس بزرگتر از opt) قرار داده میشود، بیش از 1/3 نیست.

  اثبات: ...

مسأله بسته بندی (Bin-Packing)

- **لم 9.2:** اگر opt تعداد جعبه بهینه برای یک نمونه مسأله بسته‌بندی باشد، تعداد اشیایی که توسط الگوریتم اولین جای دهی غیر افزایشی در جعبه‌های اضافی قرار داده میشود، حداکثر opt-1 است.

اثبات: چون تمام اشیاء در تعداد opt جعبه جای میگیرند، داریم: $\sum_{i=1}^{n} s_i \leq opt$

بنا بر برهان خلف، فرض میکنیم الگوریتم تقریب تعداد opt شیيء را در جعبه‌های اضافی قرار میدهد. فرض کنید z_1، z_2، ... و z_{opt} حجم این اشیاء باشد. همچنین فرض کنید tot_i کل حجمی باشد که الگوریتم در جعبه B_i قرار داده است ($1 \leq i \leq opt$). باید داشته باشیم: $tot_i + z_i > 1$

چرا که در غیر اینصورت، حجم z_i میتوانست داخل B_i قرار گیرد.

بنابراین داریم:

$$\sum_{i=1}^{n} s_i \geq \sum_{i=1}^{opt} tot_i + \sum_{i=1}^{opt} z_i = \sum_{i=1}^{opt} (tot_i + z_i) > \sum_{i=1}^{opt} 1 = opt$$

که با مطلب بیان شده در خط اول اثبات در تناقض است. این تناقض حکم را ثابت میکند.

مسأله بسته بندی (Bin-Packing)

- **قضیه 9.8:** اگر opt تعداد جعبه بهینه برای یک نمونه مسأله بسته بندی بوده و approx تعداد جعبه مورد استفاده توسط الگوریتم اولین جای دهی غیر افزایشی باشد، آنگاه خواهیم داشت: $approx \leq 1.5 \times opt$

اثبات: بنابر لم 9.1 و 9.2، الگوریتم تقریب حداکثر opt-1 شیء، هر یک با حجم حداکثر 1/3 را در جعبه‌های اضافی قرار میدهد. بنابراین حداکثر تعداد جعبه‌های اضافی از رابطه زیر بدست می‌آید:

$$\left\lceil (opt-1) \times \frac{1}{3} \right\rceil = \left\lceil \frac{opt-1}{3} \right\rceil = \frac{opt-1+k}{3}$$

که k میتواند 0، 1، یا 2 باشد. با در نظر گرفتن بیشترین مقدار برای k نتیجه میشود که تعداد جعبه‌های اضافی کمتر یا مساوی با (opt+1)/3 است. یعنی:

$$approx \leq opt + \frac{opt+1}{3} \longrightarrow \frac{approx}{opt} \leq \frac{opt + (opt+1)/3}{opt} = \frac{4}{3} + \frac{1}{3opt}$$

بیشترین مقدار سمت راست بازاء opt=1 حاصل میشود، که در این حالت الگوریتم تقریب هم بهینه عمل میکند. لذا میتوان حد بالا را با فرض opt=2 بدست آورد:

$$\frac{approx}{opt} \leq \frac{4}{3} + \frac{1}{3 \times 2} = \frac{3}{2}$$

به پایان آمد این دفتر حکایت همچنان باقی است

مژده وصل تو کو کز سر جان برخیزم
طایر قدسم و از دام جهان برخیزم

یا رب از ابر هدایت برسان بارانی
پیشتر زان که چو گردی ز میان برخیزم

√448