انشعاب و حد (Branch-and-Bound)

مقدمه و یادآوری

پس از آشنایی با روش عقب‌گرد (Backtracking)، در این جلسه به سراغ یک روش قدرتمند دیگر برای حل مسائل بهینه‌سازی، یعنی **«انشعاب و حد» (Branch-and-Bound)** می‌رویم.

ایده اصلی در الگوریتم عقب‌گرد، هرس کردن یا قطع کردن شاخه‌هایی از درخت فضای حالت بود که مشخص می‌شد به راه‌حل منجر نمی‌شوند. روش انشعاب و حد این ایده را برای **مسائل بهینه‌سازی (Optimization Problems)** تقویت و گسترش می‌دهد. مسائل بهینه‌سازی به دنبال کمینه یا بیشینه کردن یک **تابع هدف (Objective Function)**، مانند طول یک مسیر، ارزش کل اشیاء انتخابی یا هزینه یک تخصیص، تحت یک سری محدودیت‌ها هستند.

انشعاب و حد (Branch-and-Bound) چیست؟

در ادبیات مسائل بهینه‌سازی، باید بین دو مفهوم کلیدی تمایز قائل شویم:

- **راه‌حل امکان‌پذیر (Feasible Solution):** یک نقطه در فضای جستجوی مسئله که تمامی محدودیت‌های مسئله را برآورده می‌کند. (مثلاً یک دور همیلتونی در مسئله فروشنده دوره‌گرد).
- **راه‌حل بهینه (Optimal Solution):** یک راه‌حل امکان‌پذیر که بهترین مقدار ممکن را برای تابع هدف دارد. (مثلاً کوتاه‌ترین دور همیلتونی).

روش انشعاب و حد، در مقایسه با عقب‌گرد، به دو مؤلفه اضافی نیاز دارد:

1. **روشی برای محاسبه یک «حد» (Bound):** برای هر گره در درخت فضای حالت، باید بتوانیم یک حد برای بهترین مقدار تابع هدفی که از این گره قابل دستیابی است، محاسبه کنیم.
   - در مسائل **کمینه‌سازی**، این حد یک **حد پایین (Lower Bound)** است.
   - در مسائل **بیشینه‌سازی**، این حد یک **حد بالا (Upper Bound)** است.
2. **مقدار بهترین راه‌حل یافت‌شده تاکنون (Best-Solution-So-Far):** در حین اجرای الگوریتم، مقداری از بهترین راه‌حل کاملی که تا آن لحظه پیدا کرده‌ایم را نگهداری می‌کنیم.

با در اختیار داشتن این دو اطلاعات، می‌توانیم مقدار حد یک گره را با بهترین راه‌حل یافت‌شده تاکنون مقایسه کنیم. اگر حد یک گره بهتر از بهترین راه‌حل موجود نباشد (مثلاً در یک مسئله کمینه‌سازی، حد پایین یک گره از هزینه بهترین راه‌حل فعلی بیشتر باشد)، آن گره **«غیرامیدبخش» (Non-promising)** تلقی شده و کل زیردرخت آن **هرس (Pruned)** می‌شود.

چه زمانی یک گره هرس (Prune) می‌شود؟

در الگوریتم انشعاب و حد، جستجو در یک مسیر از درخت فضای حالت در هر یک از سه حالت زیر متوقف می‌شود:

1. **مقدار حد گره بهتر از بهترین راه‌حل فعلی نیست.** همان‌طور که توضیح داده شد، هیچ راه‌حلی که از این گره به دست آید، نمی‌تواند بهتر از راه‌حل بهینه‌ای باشد که قبلاً پیدا کرده‌ایم.
2. **گره نشان‌دهنده هیچ راه‌حل امکان‌پذیری نیست.** این حالت زمانی رخ می‌دهد که محدودیت‌های مسئله در مسیر رسیدن به این گره نقض شده باشند.
3. **گره نشان‌دهنده یک راه‌حل امکان‌پذیر منفرد است (یک گره برگ).** در این حالت، چون انتخاب دیگری وجود ندارد، مقدار تابع هدف برای این راه‌حل را با بهترین راه‌حل یافت‌شده تاکنون مقایسه می‌کنیم و در صورتی که بهتر بود، آن را به‌روزرسانی می‌کنیم.

مثال اول: مسئله تخصیص کار (Assignment Problem)

فرض کنید می‌خواهیم n نفر را به n کار تخصیص دهیم، به طوری که هزینه کل کمترین مقدار ممکن باشد. هزینه تخصیص هر فرد به هر کار در یک ماتریس هزینه C داده شده است. هدف، انتخاب یک درایه از هر سطر ماتریس است، به طوری که هیچ دو درایه‌ای در یک ستون نباشند و مجموع آن‌ها کمینه شود.

برای مثال، ماتریس هزینه زیر را در نظر بگیرید:

$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$

محاسبه حد پایین (Lower Bound)

یک راه ساده برای محاسبه حد پایین برای هزینه کل، جمع کردن کمترین عنصر در هر سطر ماتریس است. این مقدار قطعاً از هزینه هر راه‌حل امکان‌پذیری کمتر یا مساوی خواهد بود.

برای مثال بالا:

$$lb = 2 + 3 + 1 + 4 = 10$$

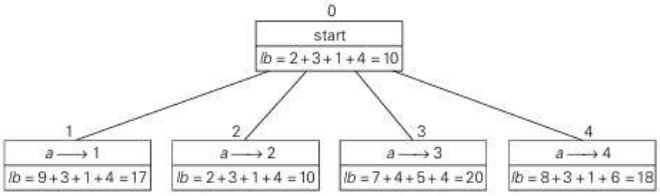
نکته مهم: این مقدار صرفاً یک حد پایین است و ممکن است هزینه یک راه‌حل امکان‌پذیر نباشد (در اینجا، مقادیر 3 و 1 هر دو از ستون سوم انتخاب شده‌اند که مجاز نیست).

استراتژی جست‌وجوی بهترین-اول (Best-First Search)

برخلاف عقب‌گرد که معمولاً از جستجوی عمق-اول (DFS) استفاده می‌کند، در انشعاب و حد اغلب از روش **«بهترین-اول»** استفاده می‌شود. در این استراتژی، ما به جای تولید تنها یک فرزند از آخرین گره امیدبخش، **تمام فرزندان امیدبخش‌ترین گره** را تولید می‌کنیم. «امیدبخش‌ترین گره» در میان تمام گره‌های زنده (برگ‌های درخت تا این لحظه)، گرهی است که بهترین حد را دارد (در مسائل کمینه‌سازی، کمترین حد پایین را دارد).

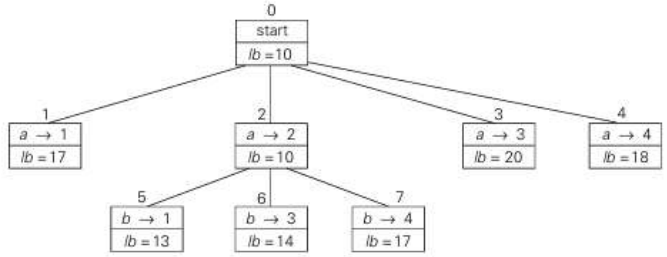
روند حل مسئله

1. **گره ریشه (گره 0):** در ابتدا هیچ تخصیصی انجام نشده است. حد پایین برابر با 10 است.
2. **سطح اول:** فرزندان ریشه را تولید می‌کنیم که معادل تخصیص یک کار به شخص `a` هستند.



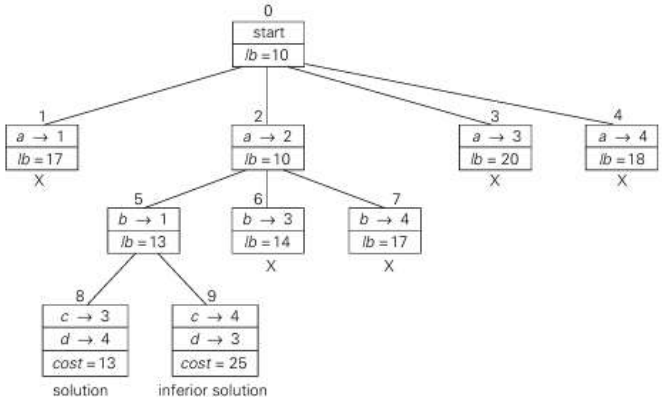
شکل ۵-۱۲ از کتاب لویتین: این شکل درخت فضای حالت را در سطوح 0 و 1 نشان می‌دهد. چهار گره فرزند برای ریشه ایجاد می‌شود که هر کدام نماینده تخصیص یکی از کارهای 1 تا 4 به شخص `a` هستند. حد پایین برای هر گره محاسبه می‌شود .برای مثال، برای گره 1 (تخصیص کار 1 به `a`)، حد پایین برابر است با $9 + 3 + 1 + 4 = 17$. گره 2 (تخصیص کار 2 به `a`) با حد پایین 10، امیدبخش‌ترین گره است.

3. **سطح دوم:** گره 2 را بسط می‌دهیم. شخص `b` می‌تواند کارهای 1، 3 یا 4 را بگیرد.



شکل ۶-۱۲ از کتاب لویتین: این شکل گسترش گره 2 را نشان می‌دهد. سه گره جدید (5، 6 و 7) ایجاد می‌شوند.
در میان تمام گره‌های زنده (1, 3, 4, 5, 6, 7)، گره 5 با حد پایین 13 ($lb = 2 + 6 + 1 + 4 = 13$) بهترین حد را دارد و امیدبخش‌ترین گره بعدی است.

1. **رسیدن به راه‌حل:** گره 5 را بسط می‌دهیم (تخصیص کار 1 به `b` پس از تخصیص کار 2 به `a`).
- **ارجاع به شکل 7-12:** این شکل درخت کامل حل مسئله را نشان می‌دهد. بسط گره 5 منجر به دو گره برگ می‌شود:
  - **گره 8:** یک راه‌حل امکان‌پذیر $a \to 2, b \to 1, c \to 3, d \to 4$ با هزینه کل **13** است. این مقدار به عنوان «بهترین راه‌حل یافت‌شده تاکنون» ذخیره می‌شود.
  - **گره 9:** یک راه‌حل امکان‌پذیر دیگر با هزینه 25 است. چون 25 > 13 است، این گره یک راه‌حل نامناسب (inferior) است.



شکل ۷-۱۲ از کتاب لویتین: این شکل درخت کامل حل مسئله را نشان می‌دهد. بسط گره 5 منجر به دو گره برگ می‌شود:

گره 8: یک راه‌حل امکان‌پذیر $a \to 2, b \to 1, c \to 3, d \to 4$ با هزینه کل **13** است. این مقدار به عنوان «بهترین راه‌حل یافت‌شده تاکنون» ذخیره می‌شود.

گره 9: یک راه‌حل امکان‌پذیر دیگر با هزینه 25 است. چون 25 > 13 است، این گره یک راه‌حل نامناسب (inferior) است.

1. **هرس کردن گره‌های باقیمانده:** اکنون به سراغ گره‌های زنده دیگر می‌رویم و حدهای پایین آن‌ها را با بهترین راه‌حل یافت‌شده (13) مقایسه می‌کنیم:
- گره 1: $lb = 17 > 13$ (هرس می‌شود)
- گره 3: $lb = 20 > 13$ (هرس می‌شود)
- گره 4: $lb = 18 > 13$ (هرس می‌شود)
- گره 6: $lb = 14 > 13$ (هرس می‌شود)
- گره 7: $lb = 17 > 13$ (هرس می‌شود)

از آنجایی که تمام گره‌های دیگر هرس شدند، راه‌حل یافت‌شده در گره 8 با هزینه 13 **راه‌حل بهینه** است.

مثال دوم: مسئله کوله‌پشتی (Knapsack Problem)

در این مسئله، با داشتن n قلم کالا با وزن‌ها و ارزش‌های مشخص، و یک کوله‌پشتی با ظرفیت معین W، هدف یافتن زیرمجموعه‌ای از کالاهاست که مجموع ارزش آن‌ها بیشینه شود و مجموع وزنشان از ظرفیت کوله‌پشتی تجاوز نکند.

برای حل با انشعاب و حد، ابتدا کالاها را بر اساس نسبت **ارزش به وزن** (v_i/w_i) به صورت نزولی مرتب می‌کنیم.

کالا	وزن (w)	ارزش (v)	ارزش/وزن
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4
ظرفیت کوله‌پشتی: W = 10			

محاسبه حد بالا (Upper Bound)

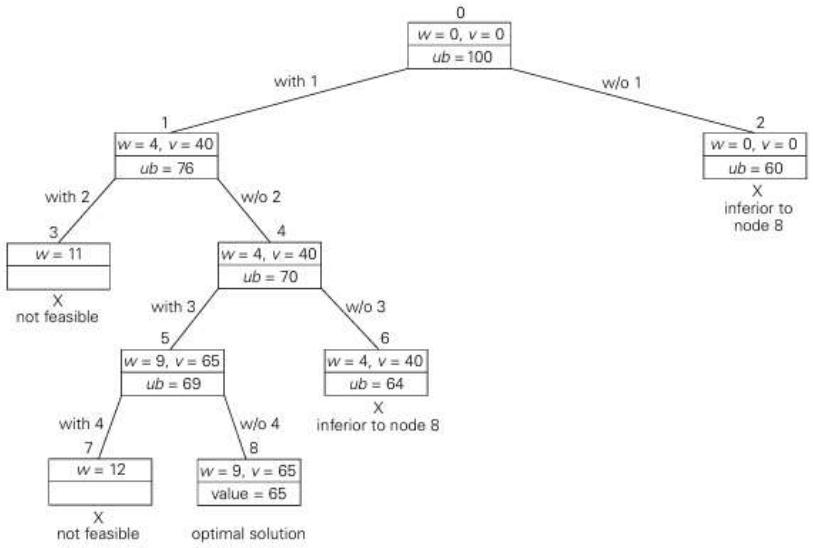
در این مسئله (یک مسئله بیشینه‌سازی)، به یک **حد بالا** نیاز داریم. برای هر گره، حد بالا را می‌توان به صورت زیر محاسبه کرد:

$$ub = v + (W - w) \times \frac{v_{i+1}}{w_{i+1}}$$

- v **و** w**:** مجموع ارزش و وزن کالاهای انتخاب‌شده تا گره فعلی.
- $W - w$**:** ظرفیت باقیمانده کوله‌پشتی.
- $\frac{v_{i+1}}{w_{i+1}}$**:** بهترین نسبت ارزش به وزن در میان کالاهای باقیمانده.

این فرمول به معنی این است که بهترین حالت ممکن، پر کردن ظرفیت باقیمانده کوله‌پشتی با «بهترین ماده ممکن» (کالایی با بیشترین چگالی ارزش) است.

روند حل مسئله



شکل ۸-۱۲ از کتاب لویتین: این شکل درخت فضای حالت مربوط به حل این نمونه را با استراتژی بهترین-اول نشان می‌دهد. درخت به صورت باینری است: شاخه چپ به معنی «شامل کالا» و شاخه راست به معنی «عدم شامل کالا» است.

1. **گره ریشه (0):** $w = 0, v = 0$. حد بالا: $ub = 0 + (10 - 0) \times 10 = 100$.
2. **سطح اول (کالای 1):**
- **گره 1 (با کالای 1):** $w = 4, v = 40$. حد بالا: $ub = 40 + (10 - 4) \times 6 = 76$.
- **گره 2 (بدون کالای 1):** $w = 0, v = 0$. حد بالا: $ub = 0 + (10 - 0) \times 6 = 60$.
- گره 1 امیدبخش‌تر است $(76 > 60)$.
3. **سطح دوم (بسط گره 1 نسبت به کالای 2):**
- **گره 3 (با کالای 2):** $w = 4 + 7 = 11$. این گره هرس می‌شود چون وزنش از ظرفیت (10) بیشتر است.
- **گره 4 (بدون کالای 2):** $w = 4, v = 40$. حد بالا: $ub = 40 + (10 - 4) \times 5 = 70$.
- در این لحظه گره‌های زنده 2 و 4 هستند. گره 4 امیدبخش‌تر است $(70 > 60)$.
4. **سطح سوم (بسط گره 4 نسبت به کالای 3):**
- **گره 5 (با کالای 3):** $w = 4 + 5 = 9, v = 40 + 25 = 65$. حد بالا: $ub = 65 + (10 - 9) \times 4 = 69$.
- **گره 6 (بدون کالای 3):** $w = 4, v = 40$. حد بالا: $ub = 40 + (10 - 4) \times 4 = 64$.
- در این لحظه، هر گره یک راه‌حل امکان‌پذیر را نشان می‌دهد. بهترین راه‌حل فعلی، گره 5 با ارزش 65 است.
5. **سطح چهارم (بسط گره 5 نسبت به کالای 4):**
- **گره 7 (با کالای 4):** $w = 9 + 3 = 12$. هرس می‌شود (اضافه وزن).
- **گره 8 (بدون کالای 4):** $w = 9, v = 65$. این یک راه‌حل کامل با ارزش 65 است. بهترین راه‌حل یافت‌شده تاکنون، 65 است.
6. **هرس گره‌های باقیمانده:**
- گره 2: $ub = 60 < 65$. هرس می‌شود.
- گره 6: $ub = 64 < 65$. هرس می‌شود.

راه‌حل بهینه: زیرمجموعه {کالای 1، کالای 3} با ارزش کل 65 است.

مثال سوم: مسئله فروشنده دوره‌گرد (Traveling Salesman Problem - TSP)

هدف، یافتن کوتاه‌ترین دوری است که از هر شهر دقیقاً یک بار عبور کند و به شهر مبدأ بازگردد.

محاسبه حد پایین (Lower Bound)

برای گراف‌های با ماتریس فاصله متقارن، می‌توان یک حد پایین به صورت زیر محاسبه کرد:

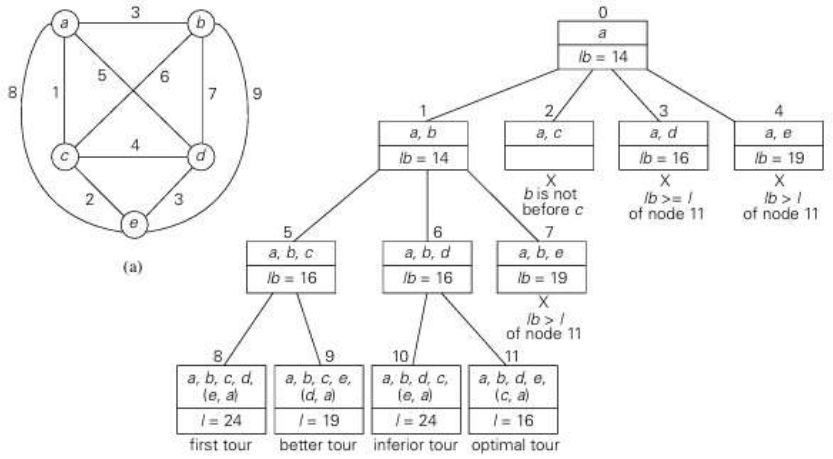
1. برای هر شهر i، مجموع فواصل آن تا دو شهر نزدیکش را پیدا کن (s_i).
2. مجموع این مقادیر را برای همه شهرها محاسبه کن ($s = \sum s_i$).
3. حد پایین برابر است با $lb = \lceil s/2 \rceil$.

ارجاع به شکل 9-12 (بخش a): این شکل یک گراف وزن‌دار با 5 رأس (شهر) را نشان می‌دهد. با استفاده از فرمول بالا برای این گراف:

- شهر a : $4 = 3 + 1$
- شهر b : $9 = 6 + 3$
- شهر c : $3 = 2 + 1$
- شهر d : $7 = 4 + 3$
- شهر e : $5 = 3 + 2$

مجموع $s = 4 + 9 + 3 + 7 + 5 = 28$.
حد پایین برای هر توری در این گراف: $lb = \lceil 28/2 \rceil = 14$.

روند حل مسئله



شکل ۹-۱۲ از کتاب لویتین: این شکل، درخت فضای حالت برای یافتن کوتاه‌ترین دور همیلتونی را نمایش می‌دهد.

- الگوریتم با شروع از رأس `a` و حد پایین 14 آغاز می‌شود.
- با اضافه کردن یال‌ها به مسیر (مثلاً رفتن از `a` به `b`)، حد پایین برای گره‌های جدید به‌روزرسانی می‌شود.
- در طی مسیر، یک تور کامل با طول 24 پیدا می‌شود. سپس تور بهتری با طول 19 یافت می‌شود. این مقدار (19) به عنوان بهترین راه‌حل فعلی ثبت می‌شود.
- گره‌هایی که حد پایین آن‌ها بیشتر یا مساوی 19 باشد، هرس می‌شوند.
- در نهایت، الگوریتم یک تور بهینه (a, b, d, e, c, a) با طول **16** پیدا می‌کند و چون حد پایین گره‌های زنده دیگر از 16 کمتر نیست، این راه‌حل به عنوان بهینه تأیید می‌شود.