Optimizations and Extensions for Weighted CFG Parsers

Richard Mörbitz

Chair for foundations of programming
Institute for theoretical computer science
TU Dresden

2021-06-21

Outline

Pruning

- Pruning for CKY parsing
- Pruning for deductive parsing

k-best parsing

A*-parsing

Pruning

- During CKY or deductive parsing many items are explored which are not part of the best derivation

Pruning

- During CKY or deductive parsing many items are explored which are not part of the best derivation
- Idea: avoid items that are not part of the best derivation to speed up parsing

Pruning

- During CKY or deductive parsing many items are explored which are not part of the best derivation
- Idea: avoid items that are not part of the best derivation to speed up parsing
- Problem: How can we know these items in advance?

Pruning

- During CKY or deductive parsing many items are explored which are not part of the best derivation
- Idea: avoid items that are not part of the best derivation to speed up parsing
- Problem: How can we know these items in advance?
- Practical solution: Use simple methods but take the risk of finding suboptimal derivation.

Pruning for CKY parsing

Consider this slightly modified (red) version of the CKY algorithm:

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 \dots t_n$ where $t_1, \dots, t_n \in \Sigma$
Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid 0 \le i < j \le n, A \in N)$ such that, for all i, j, A,
$c_{i,j,A} = \max\{\mu(d) \mid d \in D_G^A(t_{i+1} \dots t_j)\} \cup \{0\}$

function $\text{CKY}(P,\ \mu,\ t_1 \dots t_n)$
  $(c_{i,j,A} := 0 \mid 0 \le i < j \le n, A \in N)$
  for $1 \le i \le n$ **do**
    for $A \to t_i \in P$ **do**
      $c_{i-1,i,A} := \max\{c_{i-1,i,A}, \mu(A \to t_i)\}$
  for $2 \le r \le n$ **do**
    for $0 \le i \le n - r$ **do**
      $j := i + r$
      for $m \in \{i+1, i+2, \dots, j-1\}$ **do**
        for $B, C \in N$ **do**
          for $A \in N$ such that $A \to BC \in R$ **do**
            $c_{i,j,A} := \max\{c_{i,j,A}, \mu(A \to BC) \cdot c_{i,m,B} \cdot c_{m,j,C}\}$
  return c

Pruning for CKY parsing

An abstract "pruning operation" on the chart can be included (lines 6, 15).
We speed up the algorithm by skipping the application of cfg rules to chart cells with 0 probability (line 12).

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 \dots t_n$ where $t_1, \dots, t_n \in \Sigma$
Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid 0 \leq i < j \leq n, A \in N)$ such that, for all i, j, A, $c_{i,j,A} \leq \max\{\mu(d) \mid d \in D_G^A(t_{i+1} \dots t_j)\} \cup \{0\}$

1: **function** $\textsc{CKY}(P,\ \mu,\ t_1 \dots t_n)$
2: $\quad (c_{i,j,A} := 0 \mid 0 \leq i < j \leq n, A \in N)$
3: $\quad$ **for** $1 \leq i \leq n$ **do**
4: $\quad\quad$ **for** $A \to t_i \in P$ **do**
5: $\quad\quad\quad c_{i-1,i,A} := \max\{c_{i-1,i,A}, \mu(A \to t_i)\}$
6: $\quad\quad (c_{i-1,i,A} \mid A \in N) := \mathrm{prune}((c_{i-1,i,A} \mid A \in N))$
7: $\quad$ **for** $2 \leq r \leq n$ **do**
8: $\quad\quad$ **for** $0 \leq i \leq n - r$ **do**
9: $\quad\quad\quad j := i + r$
10: $\quad\quad\quad$ **for** $m \in \{i+1, i+2, \dots, j-1\}$ **do**
11: $\quad\quad\quad\quad$ **for** $B, C \in N$ **do**
12: $\quad\quad\quad\quad\quad$ **if** $c_{i,m,B} = 0$ or $c_{m,j,C} = 0$ **then** continue
13: $\quad\quad\quad\quad\quad$ **for** $A \in N$ such that $A \to BC \in R$ **do**
14: $\quad\quad\quad\quad\quad\quad c_{i,j,A} := \max\{c_{i,j,A}, \mu(A \to BC) \cdot c_{i,m,B} \cdot c_{m,j,C}\}$
15: $\quad\quad\quad (c_{i,j,A} \mid A \in N) := \mathrm{prune}((c_{i,j,A} \mid A \in N))$
16: $\quad$ **return** c

Pruning for CKY parsing

How can the pruning operation be implemented?

- Threshold beam (set all cell probabilities to 0 if worse than $\theta \cdot$ best cell probability)

Require: family $c = (c_{i,j,A} \in \mathbb{R} \mid A \in N)$, threshold $\theta \in [0, 1]$
Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid A \in N)$

function $\textsc{Prune}(c)$
  $m = \max_{A \in N}\{c_{i,j,A} \mid A \in N\}$
  for $A \in N$ **do**
    if $c_{i,j,A} < m \cdot \theta$ **then**
      $c_{i,j,A} := 0$
  return c

Pruning for CKY parsing

How can the pruning operation be implemented?

- Threshold beam (set all cell probabilities to 0 if worse than $\theta \cdot$ best cell probability)

  Require: family $c = (c_{i,j,A} \in \mathbb{R} \mid A \in N)$, threshold $\theta \in [0,1]$
  Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid A \in N)$

  function $\textsc{Prune}(c)$
      $m = \max_{A \in N}\{c_{i,j,A} \mid A \in N\}$
      for $A \in N$ **do**
          if $c_{i,j,A} < m \cdot \theta$ **then**
              $c_{i,j,A} := 0$
      return c

- Fixed-sized beam (set all but n best cell probabilities to 0)

  Require: family $c = (c_{i,j,A} \in \mathbb{R} \mid A \in N)$, size $1 \leq n \leq |N|$
  Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid A \in N)$

  function $\textsc{Prune}(c)$
      $[s_1, \ldots, s_n] = n\text{-best}\{c_{i,j,A} \mid A \in N\}$
      for $A \in N$ **do**
          if $c_{i,j,A} < s_n$ **then**
              $c_{i,j,A} := 0$
      return c

$n\text{-best}(C)$ returns list of the n highest values in C in descending order

Pruning for CKY parsing– implementation considerations

- No changes to data structures required.
- More speed-ups might be obtained by not adding items to chart which for sure would later be pruned:
  - Threshold beam: store weight m of currently best item. If new item has weight m below $\theta \cdot m$, it is save to prune immediately.
  - Fixed-size beam: store weights of the n best items. If the weight of new is below of worst item, prune immediately.

Pruning for deductive parsing

Original algorithm for deductive parsing.

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 \ldots t_n$ where $t_1, \ldots, t_n \in \Sigma$
Ensure: family $(c_{i,j,A} \colon \mathbb{R} \mid 0 \leq i < j \leq n, A \in N)$ such that
$c_{i,j,A} = \max\{\mu(d) \mid d \in D_G^A(t_{i+1} \ldots t_j)\} \cup \{0\}$

function $\textsc{deduce}(P,\ \mu,\ t_1 \ldots t_n)$
  $\mathit{queue} := \{(i-1, A, i, \mu(A \to t_i)) \mid 1 \leq i \leq n, A \to t_i \in P\}$
  $(c_{i,j,A} := 0 \mid 0 \leq i < j \leq n, A \in N)$
  while $\mathit{queue} \neq \varnothing$ **do**
    $(i, A, j, w) := \operatorname{argmax}_{(i,A,j,w) \in \mathit{queue}}\ w$
    $\mathit{queue} \setminus= \{(i, A, j, w)\}$
    if $c_{i,j,A} = 0$ **then**
      $c_{i,j,A} := w$
      $\mathit{queue} \cup= \{(i, A', j', \mu(A' \to AC) \cdot w \cdot c_{j,j',C}) \mid A' \to AC \in P\}$
      $\mathit{queue} \cup= \{(i', A', j, \mu(A' \to BA) \cdot c_{i',i,B} \cdot w) \mid A' \to BA \in P\}$
      $\mathit{queue} \cup= \{(i, A', j, \mu(A' \to A) \cdot w) \mid A' \to A \in P\}$
  return c

Pruning for deductive parsing

Pruning operation on queue is added.

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 \dots t_n$ where $t_1, \dots, t_n \in \Sigma$
Ensure: family $(c_{i,j,A} \colon \mathbb{R} \mid 0 \le i < j \le n, A \in N)$ such that
$c_{i,j,A} \le \max\{\mu(d) \mid d \in D_G^A(t_{i+1} \dots t_j)\} \cup \{0\}$

function $\textsc{deduce}(P,\ \mu,\ t_1 \dots t_n)$
  $queue := \{(i-1, A, i, \mu(A \to t_i)) \mid 1 \le i \le n, A \to t_i \in P\}$
  $(c_{i,j,A} := 0 \mid 0 \le i < j \le n, A \in N)$
  while $queue \ne \emptyset$ **do**
    $(i, A, j, w) := \operatorname{argmax}_{(i,A,j,w) \in queue}\ w$
    $queue \setminus= \{(i, A, j, w)\}$
    if $c_{i,j,A} = 0$ **then**
      $c_{i,j,A} := w$
      $queue \cup= \{(i, A', j', \mu(A' \to AC) \cdot w \cdot c_{j,j',C}) \mid A' \to AC \in P\}$
      $queue \cup= \{(i', A', j, \mu(A' \to BA) \cdot c_{i',i,B} \cdot w) \mid A' \to BA \in P\}$
      $queue \cup= \{(i, A', j, \mu(A' \to A) \cdot w) \mid A' \to A \in P\}$
      $\mathrm{prune}(queue)$
  return c

Pruning for deductive parsing

Again two options for pruning:

- Threshold beam (remove each queue item if its probability is worse than $\theta \cdot$ prob. of best queue item)

Require: set *queue* $\subseteq \mathbb{N} \times N \times \mathbb{N} \times \mathbb{R}$, threshold $\theta \in [0, 1]$
Ensure: set *queue*$'$ $\subseteq \mathbb{N} \times N \times \mathbb{N} \times \mathbb{R}$

1: **function** PRUNE(*queue*)
2: $\quad m = \max_{(i,A,j,w) \in queue} w$
3: $\quad$ **return** $\{(i, A, j, w) \in queue \mid w > \theta \cdot m\}$

Pruning for deductive parsing

Again two options for pruning:

- Threshold beam (remove each queue item if its probability is worse than $\theta \cdot$ prob. of best queue item)

  Require: set $queue \subseteq \mathbb{N} \times N \times \mathbb{N} \times \mathbb{R}$, threshold $\theta \in [0,1]$
  Ensure: set $queue' \subseteq \mathbb{N} \times N \times \mathbb{N} \times \mathbb{R}$

  function PRUNE($queue$)
  $m = \max_{(i,A,j,w) \in queue} w$
  return $\{(i,A,j,w) \in queue \mid w > \theta \cdot m\}$

- Fixed-sized beam (only keep the n most probable queue items)

  Require: set $queue \subseteq \mathbb{N} \times N \times \mathbb{N} \times \mathbb{R}$, size $n \in \mathbb{N}$
  Ensure: set $queue' \subseteq \mathbb{N} \times N \times \mathbb{N} \times \mathbb{R}$

  function PRUNE($queue$)
  $[i_1, \ldots, i_n] = n\text{-best}(queue)$ w.r.t. 4th tuple component
  return $\{i_1, \ldots, i_n\}$

Pruning for deductive parsing – implementation considerations

- Best implement queue as Min-max heap [Atk+86], because access to best and worst elements is required.

Pruning for deductive parsing – implementation considerations

- Best implement queue as Min-max heap [Atk+86], because access to best and worst elements is required.
- Again, don't add items to queue if they would be pruned immediately.

Pruning for deductive parsing – implementation considerations

- Best implement queue as Min-max heap [Atk+86], because access to best and worst elements is required.
- Again, don't add items to queue if they would be pruned immediately.
- Alternatively, one can shrink the queue only occasionally and not in each iteration of the main loop.

Pruning for deductive parsing – implementation considerations

- Best implement queue as Min-max heap [Atk+86], because access to best and worst elements is required.
- Again, don't add items to queue if they would be pruned immediately.
- Alternatively, one can shrink the queue only occasionally and not in each iteration of the main loop.

- Beware: Items for large spans are often more probable than items for small spans. Risk of pruning "good" large items in favour of "bad" small items.

Pruning for deductive parsing – implementation considerations

- Best implement queue as Min-max heap [Atk+86], because access to best and worst elements is required.
- Again, don't add items to queue if they would be pruned immediately.
- Alternatively, one can shrink the queue only occasionally and not in each iteration of the main loop.

- Beware: Items for large spans are often more probable than items for small spans. Risk of pruning "good" large items in favour of "bad" small items. (Solution: see A*-star parsing below)

Outline

Pruning

Pruning for CKY parsing

Pruning for deductive parsing

k-best parsing

A*-parsing

k-best parsing

- Problem: syntactic ambiguity, e.g., “She saw the astronomer with the telescope.”

k-best parsing

- Problem: syntactic ambiguity, e.g., "She saw the astronomer with the telescope."
  - "with the telescope" modifies "saw"

k-best parsing

- Problem: syntactic ambiguity, e.g., "She saw the astronomer with the telescope."
  - "with the telescope" modifies "saw"
  - "with the telescope" modifies "the astronomer"

k-best parsing

- Problem: syntactic ambiguity, e.g., “She saw the astronomer with the telescope.”
  - “with the telescope” modifies “saw”
  - “with the telescope” modifies “the astronomer“

  Each reading has a different parse tree.

k-best parsing

- Problem: syntactic ambiguity, e.g., “She saw the astronomer with the telescope.”
  - “with the telescope” modifies “saw”
  - “with the telescope” modifies “the astronomer“

  Each reading has a different parse tree.
- Solution: return multiple parse trees per sentence.

k-best parsing

- Problem: syntactic ambiguity, e.g., "She saw the astronomer with the telescope."
  - "with the telescope" modifies "saw"
  - "with the telescope" modifies "the astronomer"

  Each reading has a different parse tree.
- Solution: return multiple parse trees per sentence.
- Goal: given a sentence w, a PCFG G, and a positive integer k, find the k most probable derivations of G for w

k-best parsing – naïve algorithm

Similar to CKY algorithm. Uses functions:

- $\mathrm{sort}(c)$ – sorts a set c of tuples (according to 2nd component)
- $\mathrm{take}(k, \ell)$ – returns first k elements of list ℓ

Require: $k \in \mathbb{N}$, weighted binary CFG (N, Σ, S, R, μ), word $t_1 \cdots t_n$
Ensure: k most probable parse trees of PCFG for $t_1 \cdots t_n$

function $\textsc{kbest}(k,\ R,\ \mu,\ t_1, \ldots, t_n)$
  $b[i,j,A] := [\,]$ for each cell (i,j,A)
  for $i \in \{0, \ldots, n-1\}$ and $A \in N$ **do**
    $c := \{(A(t_{i+1}), w) \mid A \to t_{i+1} \text{ in } R, w = \mu(A \to t_{i+1})\}$
    $b[i, i+1, A] = \mathrm{take}(k, \mathrm{sort}(c))$
  for $z \in \{2, \ldots, n\}$ **do**
    for $i \in \{0, \ldots, n-z\}$ **do**
      $j := i + z$
      for $A \in N$ **do**
        $c := \{(A(d_1, d_2), w) \mid A \to BC \text{ in } R, m \in \{i+1, \ldots, j-1\},$
          $(d_1, w_1) \in b[i, m, B], (d_2, w_2) \in b[m, j, C],$
          $w = \mu(A \to BC) \cdot w_1 \cdot w_2\}$
        $b[i,j,A] = \mathrm{take}(k, \mathrm{sort}(c))$
  return $b[0, n, S]$

k-best parsing – implementation of merging

$$
\begin{aligned}
& c := \{(A(d_1, d_2), w) \mid A \to BC, m \in \{i+1, \ldots, j-1\}, \\
& \qquad (d_1, w_1) \in b[i, m, B], (d_2, w_2) \in b[m, j, C], \\
& \qquad w = \mu(A \to BC) \cdot w_1 \cdot w_2\} \\
& b[i, j, A] = \mathrm{take}(k, \mathrm{sort}(c))
\end{aligned}
$$

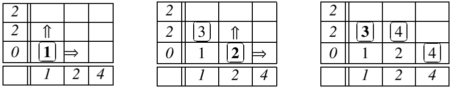
can be implemented as

$$
\begin{aligned}
& b[i, j, A] := [\,] \\
& \textbf{for } m \in \{i+1, \ldots, j-1\} \textbf{ do} \\
& \quad \textbf{for } A \to BC \text{ in } R \textbf{ do} \\
& \qquad c := \{(A(d_1, d_2), w) \mid (d_1, w_1) \in b[i, m, B], (d_2, w_2) \in b[m, j, C], \\
& \qquad\qquad w = \mu(A \to BC) \cdot w_1 \cdot w_2\} \\
& \qquad b[i, j, A] := \mathrm{mergeAndTakeK}(k, b[i, j, A], c)
\end{aligned}
$$

here $\mathrm{mergeAndTakeK}(k, \ell, c)$ returns the list of k-best items in the union of list ℓ and set c

k-best parsing – merging more efficiently [HC05]

- the set c can be computed lazily
  - avoid considering all k^2 items obtained by combining each of $b[i,m,B]$ with $b[m,j,C]$
  - instead, we only combine the best items of $b[i,m,B]$ with the best items of $b[m,j,C]$:
    - if the combination of the u-th best of $b[i,m,B]$ and v-th best of $b[m,j,C]$ was considered, then consider also the combinations $(u+1,v)$ and $(u,v+1)$



(a) (b) (c)

$b[i,j,A] := [\ ]$
for $m \in \{i+1,\dots,j-1\}$ **do**
  for $A \to BC$ in R **do**
    denote $w_{u,v} := \mu(A \to BC) \cdot w_u^1 \cdot w_v^2$ where
      $(d_u^1, w_u^1) := b[i,m,B][u]$ and
      $(d_v^2, w_v^2) := b[m,j,C][v]$
    $F := \{(1,1)\}$
    while $\max_{(u,v)\in F} w_{u,v} > \min_{(d,w)\in b[i,j,A]} w$ or $|b[i,j,A]| < k$ **do**
      $(u,v) = \operatorname{argmax}_{(u,v)\in F} w_{u,v}$
      $b[i,j,A] := \mathrm{insertAndTakeK}(k,\ \ b[i,j,A],\ \ \{(A(d_u^1,d_v^2), w_{u,v})\})$
      $F := (F \setminus \{(u,v)\}) \cup \{(u+1,v),(u,v+1)\}$

Outline

Pruning

Pruning for CKY parsing

Pruning for deductive parsing

k-best parsing

A*-parsing

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.
- How about *future costs*, i.e., the weight of $S \Rightarrow_G^* t_1 \cdots t_i \; A \; t_{j+1} \cdots t_n$?

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.
- How about *future costs*, i.e., the weight of $S \Rightarrow^*_G t_1 \cdots t_i \; A \; t_{j+1} \cdots t_n$?
- If future costs are taken into account, then maybe less items from the queue need to be processed.

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.
- How about *future costs*, i.e., the weight of $S \Rightarrow_G^* t_1 \cdots t_i \ A \ t_{j+1} \cdots t_n$?
- If future costs are taken into account, then maybe less items from the queue need to be processed.
  - Why?: Usually items with small spans are more probable than items with large spans.

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.
- How about *future costs*, i.e., the weight of $S \Rightarrow_G^* t_1 \cdots t_i \; A \; t_{j+1} \cdots t_n$?
- If future costs are taken into account, then maybe less items from the queue need to be processed.
  - Why?: Usually items with small spans are more probable than items with large spans.
  - This is counteracted by future costs which are higher for items with small spans.

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.
- How about *future costs*, i.e., the weight of $S \Rightarrow_G^* t_1 \cdots t_i \; A \; t_{j+1} \cdots t_n$?
- If future costs are taken into account, then maybe less items from the queue need to be processed.
  - Why?: Usually items with small spans are more probable than items with large spans.
  - This is counteracted by future costs which are higher for items with small spans.
- Klein and Manning [KM03] propose several admissible heuristics.

A*-parsing

- weighted deductive parsing computes for each item (i, j, A) in the chart the weight of the most probable derivation from A to $t_{i+1} \cdots t_j$.
- How about *future costs*, i.e., the weight of $S \Rightarrow_G^* t_1 \cdots t_i \; A \; t_{j+1} \cdots t_n$?
- If future costs are taken into account, then maybe less items from the queue need to be processed.
  - Why?: Usually items with small spans are more probable than items with large spans.
  - This is counteracted by future costs which are higher for items with small spans.
- Klein and Manning [KM03] propose several admissible heuristics.

- A heuristic may also be useful when pruning items during CKY parsing.

A*-parsing – Viterbi outside score

We use the admissible heuristic out:

$$\mathrm{out}(A) = \max_{d \in D_G, u,w \in \Sigma^* :\, S \overset{d}{\Rightarrow}_G uAw} \mathsf{weight}(d)$$

It can be computed by a variant of the inside/outside algorithm:

function INSIDE
  for $A \in N$ **do**
    $\mathrm{in}(A) := \max(\{\mu(A \to \alpha) \mid A \to \alpha \ \in R\} \cup \{0\})$
  while not converged **do**
    for $A \in N$ **do**
      $\mathrm{in}(A) = \max(\{\mathrm{in}(A)\} \cup \{\mu(A \to BC) \cdot \mathrm{in}(B) \cdot \mathrm{in}(C) \mid A \to BC \text{ in } R\}$
      $\cup \{\mu(A \to B) \cdot \mathrm{in}(B) \mid A \to B \text{ in } R\})$

function OUTSIDE
  set $\mathrm{out}(B) := \begin{cases} 1 & B = S \\ 0 & \text{otherwise} \end{cases}$ **for each** $B \in N$
  while not converged **do**
    for $B \in N$ **do**
      $\mathrm{out}(B) := \max(\{\mathrm{out}(B)\} \cup \{\mathrm{out}(A) \cdot \mu(A \to BC) \cdot \mathrm{in}(C) \mid A \to BC \text{ in } R\}$
      $\cup \{\mathrm{out}(A) \cdot \mu(A \to CB) \cdot \mathrm{in}(C) \mid A \to CB \text{ in } R\}$
      $\cup \{\mathrm{out}(A) \cdot \mu(A \to B) \mid A \to B \text{ in } R\})$

A*-parsing – parsing algorithm with heuristic

The out-value is now simply included when selecting the best item from the queue:

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 \dots t_n$ where $t_1, \dots, t_n \in \Sigma$

Ensure: family $(c_{i,j,A} \colon \mathbb{R} \mid 0 \leq i < j \leq n, A \in N)$ such that

$c_{i,j,A} = \max\{\mu(d) \mid d \in D_G^A(t_{i+1} \dots t_j)\} \cup \{0\}$

function $\textsc{deduce}(P,\ \mu,\ t_1 \dots t_n)$

  $queue := \{(i-1, A, i, \mu(A \to t_i)) \mid 1 \leq i \leq n, A \to t_i \in P\}$

  $(c_{i,j,A} := 0 \mid 0 \leq i < j \leq n, A \in N)$

  while $queue \neq \varnothing$ **do**

    $(i, A, j, w) := \operatorname{argmax}_{(i,A,j,w) \in queue}\ w \cdot \mathrm{out(A)}$

    $queue \setminus= \{(i, A, j, w)\}$

    if $c_{i,j,A} = 0$ **then**

      $c_{i,j,A} := w$

      $queue \cup= \{(i, A', j', \mu(A' \to AC) \cdot w \cdot c_{j,j',C}) \mid A' \to AC \in P\}$

      $queue \cup= \{(i', A', j, \mu(A' \to BA) \cdot c_{i',i,B} \cdot w) \mid A' \to BA \in P\}$

      $queue \cup= \{(i, A', j, \mu(A' \to A) \cdot w) \mid A' \to A \in P\}$

  return c

Pruning for deductive parsing – in the context of CKY

Can A*-parsing be applied for CKY parsing?
Yes: in combination with pruning:

- Threshold beam

  Require: family $c = (c_{i,j,A} \in \mathbb{R} \mid A \in N)$, threshold $\theta \in [0, 1]$
  Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid A \in N)$

  function $\textsc{Prune}(c)$
  $\quad m = \max_{A \in N}\{c_{i,j,A} \cdot \mathrm{out}(A) \mid A \in N\}$
  $\quad$ **for** $A \in N$ **do**
  $\quad\quad$ **if** $c_{i,j,A} \cdot \mathrm{out}(A) < m \cdot \theta$ **then**
  $\quad\quad\quad c_{i,j,A} := 0$
  $\quad$ **return** c

Pruning for deductive parsing – in the context of CKY

Can A*-parsing be applied for CKY parsing?
Yes: in combination with pruning:

- Threshold beam

  Require: family $c = (c_{i,j,A} \in \mathbb{R} \mid A \in N)$, threshold $\theta \in [0, 1]$
  Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid A \in N)$

  function $\textsc{Prune}(c)$
      $m = \max_{A \in N}\{c_{i,j,A} \cdot \mathrm{out}(A) \mid A \in N\}$
      for $A \in N$ **do**
          if $c_{i,j,A} \cdot \mathrm{out}(A) < m \cdot \theta$ **then**
              $c_{i,j,A} := 0$
      return c

- Fixed-size beam

  Require: family $c = (c_{i,j,A} \in \mathbb{R} \mid A \in N)$, size $1 \leq n \leq |N|$
  Ensure: family $(c_{i,j,A} \in \mathbb{R} \mid A \in N)$

  function $\textsc{Prune}(c)$
      $[s_1, \ldots, s_n] = n\text{-best}\{c_{i,j,A} \cdot \mathrm{out}(A) \mid A \in N\}$
      for $A \in N$ **do**
          if $c_{i,j,A} \cdot \mathrm{out}(A) < s_n$ **then**
              $c_{i,j,A} := 0$
      return c

[Atk+86] M. D. Atkinson et al. "Min-max Heaps and Generalized Priority Queues". In: *Commun. ACM* 29.10 (Oct. 1986), pp. 996–1000. ISSN: 0001-0782. DOI: 10.1145/6617.6621. URL: http://doi.acm.org/10.1145/6617.6621.

[HC05] Liang Huang and David Chiang. "Better k-best parsing". In: *Proceedings of the Ninth International Workshop on Parsing Technology*. Association for Computational Linguistics. 2005, pp. 53–64.

[KM03] Dan Klein and Christopher D. Manning. "A* Parsing: Fast Exact Viterbi Parse Selection". In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2003, pp. 119–126. URL: https://www.aclweb.org/anthology/N03-1016.