DCC IEEE

Presented at the 2018 IEEE
Data Compression Conference
Snowbird, Utah, USA — March 27–30

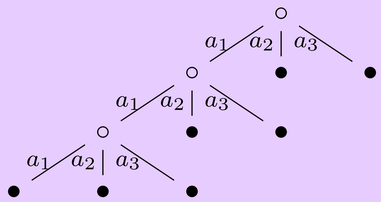
Optimal Single- and Multiple-Tree Almost Instantaneous Variable-to-Fixed Codes

Danny Dubé
Danny.Dube@ift.ulaval.ca

Fatma Haddad
Fatma.Haddad.1@ulaval.ca

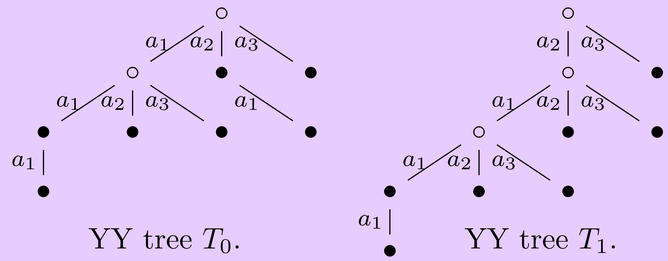
Université Laval
Canada

Tunstall Trees (PFVF)



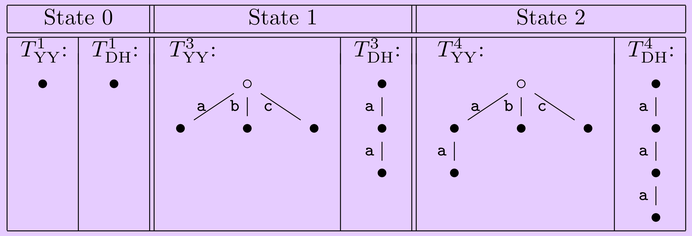
Tunstall tree T_{Tun}

Yamamoto-Yokoo Trees (AIVF)

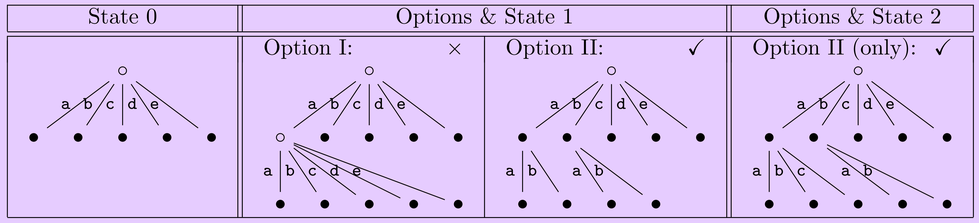


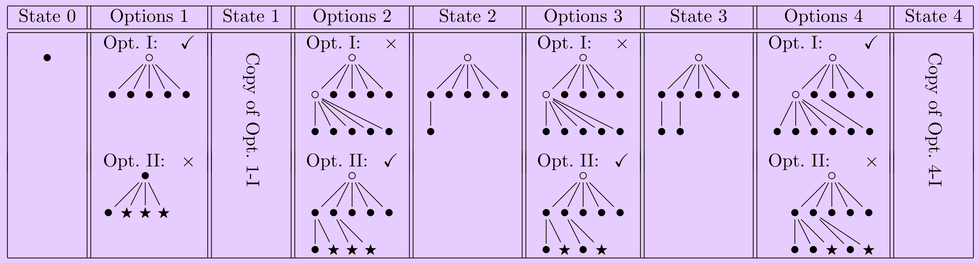
YY tree T_0. YY tree T_1.

Defect in YY Due to the Complete-Root Constraint

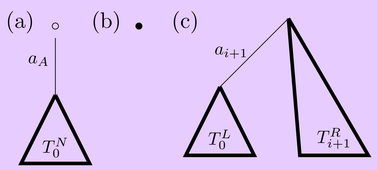


Defect in YY Due to the Full Execution of Option II





Basic Shapes of Trees Built using DP



(a) $T^N_{A-1} = \textsc{Default}(T^N_0)$
(b) $T^1_i = \textsc{Root}$, where: $i \leq A-2$
(c) $T^N_i = T^L_0 \oplus T^R_{i+1}$, where: $i \leq A-2$, $2 \leq N = L + R$

(**Figure 1.**)

Future Work

- We should verify whether applying our correctives on YY would make it optimal.
- The AI property remains a constraint on the considered VF codes, even if it is looser than the PF property, and we should investigate on the opportunities offered by the removal or relaxation of this constraint; e.g., codes with a longer delay [3].
- Related work: Almost instantaneous fixed-to-variable (AIFV) codes by Yamamoto et al [2, 3].

References

[1] S.-L. Chen and M. J. Golin. A dynamic programming algorithm for constructing optimal "1"-ended binary prefix-free codes. *IEEE Transactions on Information Theory*, 46(4):1637–1644, 2000.

[2] K. Iwata and H. Yamamoto. A dynamic programming algorithm to construct optimal code trees of AIVF codes. In *Proceedings of the International Symposium on Information Theory and Applications*, Nov. 2016.

[3] K. Iwata and H. Yamamoto. An iterative algorithm to construct optimal binary AIFV-m codes. In *Proceedings of the IEEE Information Theory Workshop*, pages 519–523, Nov. 2017.

[4] Shmuel T. Klein and Dana Shapira. Improved variable-to-fixed length codes. In *Proceedings of the Conference on String Processing and Information Retrieval*, volume 5280 of *Lecture Notes in Computer Science*, pages 39–50. Springer Berlin Heidelberg, 2009.

[5] Shmuel T. Klein and Dana Shapira. The string-to-dictionary matching problem. In *Proceedings of the Data Compression Conference*, pages 143–152, Mar. 2011.

[6] Shmuel T. Klein and Dana Shapira. The string-to-dictionary matching problem. *The Computer Journal*, 55(11):1347–1356, Nov. 2012.

[7] S. A. Savari. Variable-to-fixed length codes and plurally parsable dictionaries. In *Proceedings of the Data Compression Conference*, pages 453–462, Mar. 1999.

[8] B. P. Tunstall. *Synthesis of Noiseless Compression Codes*. PhD thesis, Georgia Institute of Technology, 1967.

[9] H. Yamamoto and H. Yokoo. Average-sense optimality and competitive optimality for almost instantaneous VF codes. *IEEE Transactions on Information Theory*, 47(6):2174–2184, Sep. 2001.

[10] Satoshi Yoshida and Takuya Kida. An efficient algorithm for almost instantaneous VF code using multiplexed parse tree. In *Proceedings of the Data Compression Conference*, pages 219–228, Mar. 2010.

[11] Satoshi Yoshida and Takuya Kida. Analysis of multiplexed parse trees for almost instantaneous VF codes. In *Proceedings of the IIAI International Conference on Advanced Applied Informatics*, pages 36–41. IEEE, Sep. 2012.

Algorithms

Algorithm COMPLETE(t)**: add a node's missing children**

$\mathcal{V} \leftarrow \{\pi(n) \mid n$ is an incomplete nodes in $t\}$
$v_{\max} \leftarrow \arg\max_{v \in \mathcal{V}} p(v)$
$\mathcal{W} \leftarrow \{v_{\max}\} \cdot \mathcal{A}$ /* Paths to children of $v_{\max}$ */
return $t + \mathcal{W}$

Algorithm TUNSTALL(M)**: build an up-to-M-codeword tree**

Require: $M \geq A$
$t_{\text{new}} \leftarrow \textsc{Root} + \{a_1, \ldots, a_A\}$
repeat
  $t_{\text{old}} \leftarrow t_{\text{new}}$
  $t_{\text{new}} \leftarrow$ COMPLETE(t_{old})
until $\#t_{\text{new}} > M$ /* Ultimate t_{new} gets discarded */
return t_{old} /* $M - (A-1) < \#t_{\text{old}} < M$ */

Algorithm EXTEND(t)**: add the best child**

$\mathcal{U} \leftarrow \{\pi(n) \mid$ node n in $t\}$ /* Paths to all nodes */
$\mathcal{V} \leftarrow \mathcal{U} \cdot \mathcal{A}$ /* Path extensions */
$\mathcal{W} \leftarrow \mathcal{V} - \mathcal{U}$ /* Paths to potential children */
$w_{\max} \leftarrow \arg\max_{w \in \mathcal{W}} p(w)$
return $t + \{w_{\max}\}$

Algorithm YY(i, M)**: build an M-codeword T_i**

Require: $0 \leq i \leq A-2$ **and** $M \geq A - i$
$t_{\text{new}} \leftarrow \textsc{Root} + \{a_{i+1}, \ldots, a_A\}$
repeat
  $t_{\text{old}} \leftarrow t_{\text{new}}$
  $t_{\text{I}} \leftarrow$ COMPLETE(t_{old}) /* Option I */
  $t_{\text{II}} \leftarrow \text{EXTEND}^{\#t_{\text{I}} - \#t_{\text{old}}}(t_{\text{old}})$ /* Option II */
  $t_{\text{new}} \leftarrow$ the best of t_{I} and t_{II}
until $\#t_{\text{new}} > M$ /* Ultimate t_{new} gets discarded */
return $\text{EXTEND}^{M - \#t_{\text{old}}}(t_{\text{old}})$ /* Option II */

Algorithm BUILD(i, N)**: build an N-codeword T_i in a DP way**

Require: $0 \leq i \leq A-1$ **and** $N \geq 1$
if $i = A - 1$ **then**
  return $\textsc{Default}(T_0^N)$ /* Fig. 1(a) */
else if $N = 1$ **then**
  return $\textsc{Root}$ /* Fig. 1(b) */
else
  $\mathcal{T} \leftarrow \{T_0^L \oplus T_{i+1}^R \mid L + R = N,\ L \geq 1,\ R \geq 1\}$ /* Fig. 1(c) */
  return $\arg\max_{t \in \mathcal{T}}$ AvgParsewordLen(t) /* $O(N)$-time case */
end if

Algorithm FILL(M)**: build all up-to-M-codeword trees in a DP way**

Require: $M \geq 1$ /* $O(A \cdot M^2)$-time algorithm */
for $N = 1$ to M **do**
  for $i = 0$ to $A - 1$ **do**
    $T_i^N \leftarrow$ BUILD(i, N)
  end for
end for

Algorithm $\text{BUILD}^{\text{single}}$($i$, N)**: build a complete-root N-codeword T_i**

Require: $0 \leq i \leq A-1$ **and** $N \geq A - i$
if $i = A - 1$ **then**
  return $\textsc{Default}(T_0^N)$
else
  $\mathcal{S} \leftarrow \{T_0^L \oplus S_{i+1}^R \mid L + R = N,\ L \geq 1,\ R \geq 1,\ R \geq A - (i+1)\}$
  return $\arg\max_{t \in \mathcal{S}}$ AvgParsewordLen(t)
end if

Algorithm $\text{FILL}^{\text{single}}$($M$)**: build all complete-root up-to-M-codeword trees**

Require: $M \geq 1$
for $N = 1$ to M **do**
  for $i = \max(A - N, 0)$ to $A - 1$ **do**
    $S_i^N \leftarrow \text{BUILD}^{\text{single}}(i, N)$ /* S_i^N undefined for $N < A - i$ */
  end for
end for