10

Code-tree sets for efficient versatile lossless encoding

Enhancing general data compression

Abstract

Data compression is one of the most important techniques supporting modern communication. Efficient compression saves memory resources and increases data traffic capability. Our goal is to enhance compression efficiency in various uses and realize more comfortable communication. Data compression is based on encoding rules that convert values or words into binary sequences. The rule is often represented as a decision tree called a code tree. Our work focuses on a code forest, a combination of code trees, and presents an encoding-rule design realizing higher efficiency in general purpose. Generally, optimal code-forest design is not practical because it requires simultaneous optimization of code trees. The conventional method avoids this problem but instead limits the condition where it can enhance efficiency. Our method enables high-efficiency compression for a broader range of data by decomposing the optimization using predetermined shapes of code trees.

Lossless data compression

- **Data like pictures, audio, and text are stored as binary codes.**
- **Lossless compression aims at precisely representing data by shorter codes.**
- **The key is to design a rule that assigns shorter codes to frequently-appearing patterns.**

Data (*e.g.* text)

$abacabbabadaaac$

Code① : 0001001000010100010011000000010

Code② : 0100110010100100111000110

Encoding rule①

$a \leftrightarrow 00$ $b \leftrightarrow 01$

$c \leftrightarrow 10$ $d \leftrightarrow 11$

Encoding rule②

$a \leftrightarrow 0$ $b \leftrightarrow 10$

$c \leftrightarrow 110$ $d \leftrightarrow 111$

Rule② gives shorter code☺

Code “tree” and code “forest”

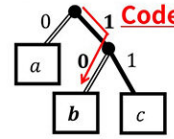
- **Code trees can represent encoding rules.**

Encoding rule

$a \leftrightarrow 0$

$b \leftrightarrow 10$

$c \leftrightarrow 11$



- **Code forests can represent more efficient rules.**

Encoding rule

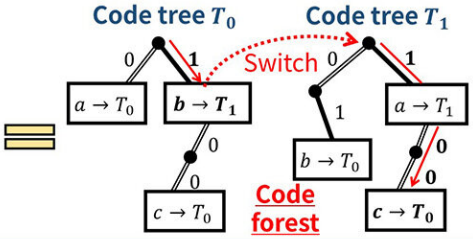
$a \leftrightarrow 0$

$ba \leftrightarrow 1\,1$

$bb \leftrightarrow 1\,01$

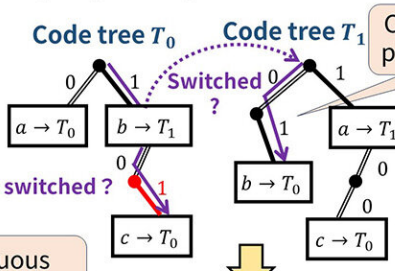
$bc \leftrightarrow 1\,100$

⋮



- **Code forests need to be properly designed to avoid outputting ambiguous codes.**
- **Forest design is difficult because whether a code tree is proper depends on the other trees.**

Unless properly designed ⋯



Code tree T_0 is proper if it is 0.

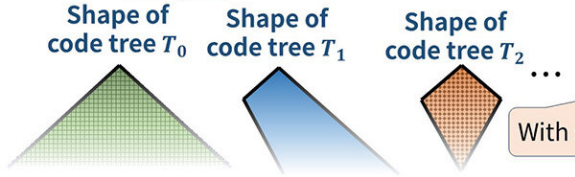
We cannot know only from T_0 whether it is proper☹

ambiguous codes occur☹

Code **101** → c ? / bb ?

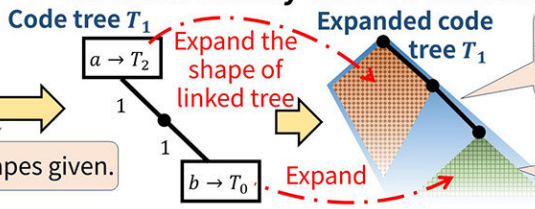
Our method: Predetermining “shapes”

- **Our method predetermines all the code trees’ shapes as constraints.**



With shapes given.

- **Predetermined shapes allow code construction independent from the other trees’ details.**
- **We theoretically derived sufficient shapes to use.**



Need only to check if the expanded tree meets its shape.

Expansion needs no other trees’ details☺

References

[1] R. Sugiura, Y. Kamamoto, T. Moriya, “General form of almost instantaneous fixed-to-variable-length codes,” *IEEE Transactions on Information Theory*, Vol. 69, No. 12, pp. 7672-7690, 2023.

[2] R. Sugiura, M. Nishino, N. Yasuda, Y. Kamamoto, T. Moriya, “Optimal construction of N-bit-delay almost instantaneous fixed-to-variable-length codes,” under review.

Contact

Ryosuke Sugiura, Linguistic Intelligence Research Group, Innovative Communication Laboratory