# A note on Prüfer-like coding and counting forests of uniform hypertrees

Christian Lavault

# A note on Prüfer-like coding and counting forests of uniform hypertrees

Christian Lavault*

**Abstract**

This note presents an encoding and a decoding algorithms for a forest of (labelled) rooted uniform hypertrees and hypercycles in linear time, by using as few as $n-2$ integers in the range $[1, n]$. It is a simple extension of the classical Prüfer code for (labelled) rooted trees to an encoding for forests of (labelled) rooted uniform hypertrees and hypercycles, which allows to count them up according to their number of vertices, hyperedges and hypertrees. In passing, we also find Cayley's formula for the number of (labelled) rooted trees as well as its generalisation to the number of hypercycles found by Selivanov in the early 70's.

**Key words:** Hypergraph, Forest of (labelled rooted) hypertrees, Prüfer code, Encoding-decoding, $b$-uniform, Enumeration.

## 1 Notations and definitions

A *hypergraph* $\mathcal{H}$ is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ denotes the set of vertices and $\mathcal{E}$ is a family of subsets of $\mathcal{V}$ each of size $\geq 2$ called hyperedges (see e.g. [1]).
Two vertices are *neighbours* if they belong to the same hyperedge. The *degree* of a vertex is the number of its neighbours. A *leaf* is a set of $b-1$ non-distinguished vertices of degree $(b-1)$ belonging to the same hyperedge.
A *hyperpath* (path) between two vertices $u$ and $v$ is a finite sequence of hyperedges $e_1, \ldots, e_k$, such that $e_i \cap e_{i+1} \neq \emptyset$ for any $1 \leq i \leq k-1$, with $u \in e_1$ and $v \in e_k$.

A hypergraph $\mathcal{H}$ is *connected* if there exists a path between any two vertices of $\mathcal{H}$. A connected hypergraph is also called a *connected component*, or simply a *component*.

A hypergraph is called *b-uniform* (or uniform) if every hyperedge $e \in \mathcal{E}$ contains exactly $b$ vertices $(2 \leq b \leq n)$ [8, 14]. For example, 2-uniform hypergraphs are simply graphs. In the present note, only connected $b$-uniform hypergraphs $(2 \leq b \leq n)$ are considered.

*LIPN (UMR CNRS 7030), Université Paris 13 99, av. J.-B. Clément 93430 Villetaneuse (France).
E-mail: lavault@lipn.univ-paris13.fr

The *excess* of a connected hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is defined as

$$exc(\mathcal{H}) = \sum_{e \in \mathcal{E}} (|e| - 1) - |\mathcal{V}|$$

(see e.g. [8, 14]). Thus, if $\mathcal{H}$ is $b$-uniform, its excess is $(b-1)|\mathcal{E}| - |\mathcal{V}|$. A *hypertree* is a component of excess $-1$, which is the smallest excess possible for a connected hypergraph $\mathcal{H}$, and hence, $exc(\mathcal{H}) \geq -1$ for any $\mathcal{H}$ (see the above definition).
A component is *rooted* if one of its vertices is distinguished from all others. A hypergraph is called a *forest* if all its components have excess $-1$ (i.e. are hypertrees), and similarly a *hypercycle* has excess 0.

# 2 Bijective enumeration of a forest of hypertrees

## 2.1 State of the art and motivations

Concerning connected graphs (2-uniform hypergraphs), there exist several methods for counting trees (see e.g. [2, 3, 4, 7, 9, 10, 11]), including of course the Prüfer code. Prüfer sequences were first introduced by Heinz Prüfer to prove Cayley's enumeration formula in 1918 [13]. In his very elegant proof [1], Prüfer verified Caley's Theorem [3] by establishing a one-to-one correspondence between labelled free trees of order $n$ and all sequences of $n-2$ positive integers from 1 to $n$. The Prüfer codes can thus be generated by a simple iterative algorithm (see also [9, vol. 1, chap. 2] and [6]).

**Remark 1.** *To compute the Prüfer sequence Seq(T) for a labelled tree $T$, iteratively delete the leaf with smallest label and append the label of its neighbor to the sequence. After $n-2$ iterations a single edge remains and we have produced a sequence Seq(T) of length $n-2$.*

Since the introduction of Prüfer codes, a linear time algorithm for its computation was given for the first time only in the 70's [12, 15], and has been later rediscovered several times in various forms [2, 4, 5]. Recently for example, the sequential encoding and decoding schemes presented in [2]. Both require an optimal $\Theta(n)$ time when applied to rooted $n$-node trees, and provide the first optimal linear time decoding algorithm for Neville's codes [2].

Amongst the most recent results, "Prüfer-like" encoding-decoding algorithms are generalizing the Prüfer code to the case of hypertrees (uniform or arbitrary). In 2009, S. Shannigrahi, S.P. Pal have shown in [17] that uniform hypertrees can be Prüfer-like encoded (and decoded) in optimal linear time $\Theta(n)$, using only $n-2$ integers in the range $[1, n]$ (see Prüfer's Theorem in [13]).

[1] Prüfer's Theorem is as follows. *There are $n^{n-2}$ sequences (called Prüfer sequence or Prüfer codes) of length $n-2$ with entries being from natural numbers; we establish a bijection between the set of trees and this set of sequences.*

In the case when hypertrees are arbitrary (non uniform), the same authors' encoding and decoding algorithms in [17] require codes of length $(n-2)+p$, where $p$ is the number of vertices belonging to more than one hyperedge, or *pivots*. Since the number $p$ of pivots is bounded by $|\mathcal{E}|$, at most $(n-2)+|\mathcal{E}|$ integers are needed to encode a general hypertree. Therefore, the design of efficient Prüfer-like encoding and decoding algorithms can be extended to arbitrary hypertrees where each hyperedge has at least two vertices. Up until now, no better bounds are known for the length of Prüfer-like codes for arbitrary hypertrees. By contrast, the exact number of distinct hypertrees is known to be $\sum_{i=0}^{n-1} \left\{ {n-1 \atop i} \right\} n^{i-1}$, where $\left\{ {p \atop q} \right\}$ denotes the Stirling numbers of the second kind [9].

The main motivation of the present note comes from analytic and bijective combinatorics of hypergraphs, including the enumeration of (labelled) rooted forests of uniform hypertrees and hypercycles [14]. These enumeration results are actually tightly linked to Prüfer-like coding and decoding of such combinatorial structures.

The following two algorithms code and decode forests of rooted uniform hypertrees, which in turns allows to enumerate these structures bijectively by using a generalization of the Prüfer sequences. The knowledge of the number of forests of rooted uniform hypertrees provides a concise and simple description of the structures, e.g. by using a recursive pruning of the leaves in the forests.

## 2.2 Encoding and decoding algorithms for a forest of (labelled) rooted hypertrees

**Definition 1.** *The forests $\mathcal{F}$ composed of $(k+1)$ (labelled) rooted $b$- uniform hypertrees, with $n$ vertices and $s$ hyperedges, is coded with a 4-tuple $(R, r, \mathcal{P}, N)$ defined as follows:*

- *$R$ is a set of $(k+1)$ vertices (roots) with distinct labels in $[n] \equiv \{1, \ldots, n\}$,*
- *$r \in R$ is one the $(k+1)$ roots,*
- *$\mathcal{P}$ is a partition of $[n] \setminus R$ into $s$ subsets, each of size $(b-1)$ and*
- *$N$ is an $(s-1)$-tuple in $[n]^{s-1}$.*

Note that the above positive integers $n \equiv n(s)$, $s$ and $k$ meet the condition

$$n = s(b-1) + k + 1,$$

and since $\mathcal{F}$ is $b$-uniform, $exc(\mathcal{F}) = s(b-1) - n$. So, $|\mathcal{E}| = s$ and $|\mathcal{V}| = n$.

Algorithm 1 is coding a given forest $\mathcal{F}$ of $k+1$ (labelled) rooted uniform hypertrees as input, and returns the 4-tuple $(R, r, \mathcal{P}, N)$ coding $\mathcal{F}$ as output.
The 4-tuple $(R, r, \mathcal{P}, N)$ is obtained from the coding in Definition 1 of a forest $\mathcal{F}$ (Algorithm 1) as follows.

1. the number of its components (i.e. the number $|R|$ of its roots),

**Algorithm 1** Encoding a forest of rooted hypertrees.

**Input:** A forest $\mathcal{F}$ of $k+1$ (labelled) rooted $b$-uniform hypertrees, with $s$ hyperedges and $n = s(b-1)+k+1$ vertices.

**Output:** The coding $(R, r, \mathcal{P}, N)$ of $\mathcal{F}$ as in Definition 1.

1. $(R, r, \mathcal{P}, N) \leftarrow (\{\text{root}\}, r, \{\,\}, (\,))$
2. **Repeat**
3. Add the set of vertices corresponding to the smallest leaf (with respect to the lexicographical order) to the partition $\mathcal{P}$.
4. Put the vertex linking that set into the $(s-1)$-tuple $N$.
5. Take $\mathcal{F}$ as the "new" forest not having the vertices corresponding to the smallest leaf.
6. **Until** there is no hyperedge remaining in $\mathcal{F}$.
7. The last vertex in $N$ is necessarily a root and $r$ is redefined as this last vertex.
8. **Return** $(R, r, \mathcal{P}, N)$.

2. the unique root vertex $r$ attached to the leaf of the last hyperedge in the pruning process,
3. the number of hyperedges $|N|+1$, and finally,
4. the number of hypertrees that are not reduced to their roots, namely the number of distinct roots in the pair $(N, r)$.

**Example**.
The forest $\mathcal{F} = (\mathcal{V}, \mathcal{E})$ of rooted uniform hypertrees depicted in Fig. 1 is as follows:
$\mathcal{V} = \{1, 2, \ldots, 22\}$ and
$\mathcal{E} = \{\{1,21,22\},\{2,17,18\},\{3,13,19\}\{4,8,18\},\{4,12,14\},\{6,7,13\},\{7,20,21\},$
$\{10,13,15\},\{11,18,21\}\}$.
The roots of the hypertrees are 5, 9, 13 and 16.

In this example, Algorithm 1 outputs ($R$, $r$, $\mathcal{P}$, $N$) s.t.
$R = \{5,9,13,16\}$,
$r = 13$,
$\mathcal{P} = \{\{1,22\},\{2,17\},\{3,19\},\{4,8\},\{6,7\},\{10,15\},\{11,18\},\{12,14\},\{20,21\}\}$ and
$N = (21,18,13,13,4,18,21,7)$.

Next, the following Algorithm 2 decodes a given 4-tuple $(R, r, \mathcal{P}, N)$ as input, and returns a forest of (labelled) rooted $b$-uniform hypertrees as output.
Within the loop of Algorithm 2, the forest $\mathcal{F}$ is found with no ambiguity by choosing the smallest leaf in the lexicographical order.

**Remark 2.** *Encoding Algorithm 1 and decoding Algorithm 2 are both running in optimal linear time, by using as few as* $n-2$ *integers in the range* $[1, n]$. *This complexity result is a direct consequence of the Prüfer-like encoding algorithm designed in [17] (see Prüfer's Theorem in [13]). The algorithm computes the hyperedge partial order on the hyperedges*

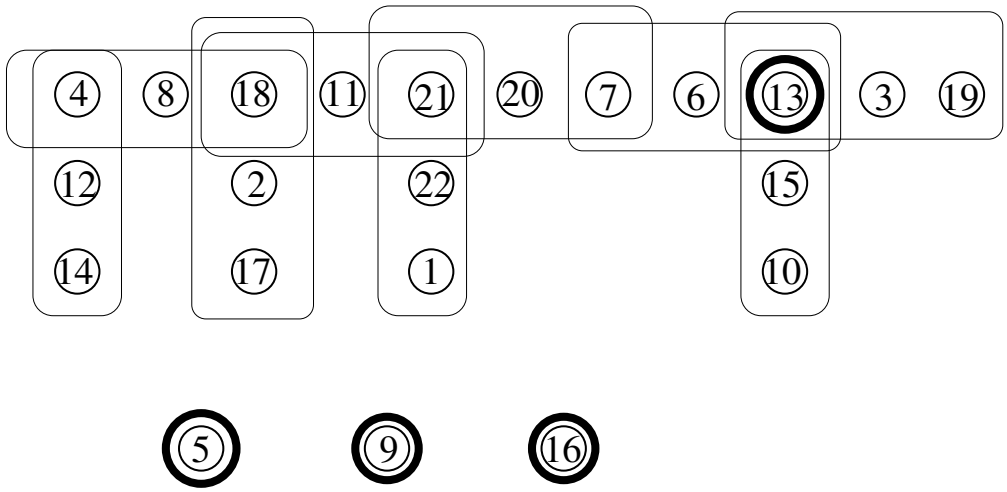


FIGURE 1: A forest of 4 (labelled) rooted hypertrees.

*of the hypertree in linear time by defining a directed acyclic graph (DAG) with vertex set $\mathcal{E}$, where each vertex represents a hyperedge of $\mathcal{H}$. (The proof is completed in [17, Lemma 2].)*

### 2.3 Enumeration of forests, hypertrees and trees

From Algorithm 1, we obtain the enumeration of forests with $(k+1)$ (labelled) rooted uniform hypertrees and $s$ hyperedges.

**Theorem 1.** *The number of forests having $(k+1)$ (labelled) rooted $b$-uniform hypertrees and $s$ hyperedges is*

$$\binom{n}{k+1}(k+1)\left[\frac{(n-k-1)!}{s!\,(b-1)!^s}\right]n^{s-1} \;=\; \frac{n!}{k!}\,\frac{n^{s-1}}{s!\,(b-1)!^s}\,, \tag{1}$$

*where the number of vertices is $n \equiv n(s) = s(b-1)+k+1$.*

*Proof.* The proof stems directly from the one-to-one correspondence constructed in Algorithm 1, which codes forests $\mathcal{F}$ with the set of 4-tuples $(R, r, \mathcal{P}, N)$ defined as in Definition 1. Indeed, the number of forests of $(k+1)$ (labelled) rooted $b$-uniform hypertrees and $s$ hyperedges is equal to $|R| \times \#roots \times |\mathcal{P}| \times |N|$.

Now, we have $|R| = \binom{n}{k+1}$, the number of roots is $(k+1)$ , $|\mathcal{P}| = \dfrac{(n-k-1)!}{s!\,(b-1)!^s}$ and $|N| = n^{s-1}$. So, after simplifications, Theorem 1 follows. □

Setting $k = 0$ in the above Eq. (1) (Theorem 1) yields $n!\,\dfrac{n^{s-1}}{s!(b-1)!^s}$. Whence the following

**Algorithm 2** Decoding to a forest of rooted hypertrees.

**Input:** Integers $n, k$ and $s$ meeting the condition $n = s(b-1)+k+1$ and the coding $(R, r, \mathcal{P}, N)$ of $\mathcal{F}$, as in Definition 1.

**Output:** Forest $\mathcal{F}$ of rooted $b$-uniform hypertrees, whose $k+1$ roots are the vertices of $R$.

1. **Repeat**
2. Build one hyperedge with the first vertex in the $(s-1)$-tuple $N$ and the vertices of the first set of $\mathcal{P}$ (w.r. to the lexicographic order) having no vertex still in the remaining set $N$.
3. Remove the above set from $\mathcal{P}$ and delete the first vertex from the $(s-1)$-tuple $N$.
4. **Until** the set $N$ is empty.
5. Build the $(s-1)$-th hyperedge with the last subset of $\mathcal{P}$ and the vertex $r$.
6. **Return** the forest $\mathcal{F}$ obtained.

**Corollary 1.** *The number of (labelled) rooted $b$-uniform hypertrees with $s$ hyperedges is*

$$\frac{(n-1)!}{s!\,(b-1)!^s}\,n^s,$$

*where the number of vertices is $n \equiv n(s) = s(b-1)+1$.*

Note that, whenever $b = 2$ (and thus $s = n-1$), Corollary 1 is a generalization of Cayley's Theorem enumerating rooted trees of $n$ vertices: *for any $n \geq 1$, the number of (nonplane labelled) rooted trees of $n$ vertices is $n^{n-1}$* [3].

One of the advantages offered by a bijective construction proof is also the possibility of performing a random generation and learn some characteristic properties of the structures in $\mathcal{F}$ [14].

A generalization of Subsection 2.3 also gives an explicit expression of the number of uniform hypercycles and obtain an alternative proof of Selivanov's 1972 enumeration result.

## 3 Enumeration of uniform hypercycles

Together with hypertrees, hypercycles are the simplest structures and they have excess 0. Along the same lines as in Theorem 1, the following bijective proof gives the enumeration formula (first given by Selivanov in [16]) of the uniform hypercycles in a forest $\mathcal{F}$.

**Theorem 2.** *[16] The number of $b$-uniform hypercycles with $s$ hyperedges is*

$$\left(\frac{(b-1)n!\,n^{s-1}}{2\,(b-1)!^s}\right)\sum_{j=2}^{s}\frac{j}{s^j(s-j)!} \;=\; \left(\frac{(b-1)n!\,n^{s-1}}{2\,(b-1)!^s}\right)\frac{1}{s(s-2)!},$$

*where the number of vertices is $n \equiv n(s) = s(b-1)$.*

*Proof.* A hypercycle having a cycle length $j$ corresponds to a forest $\mathcal{F}$ of $j(b-1)$ rooted hypertrees, up to an arrangment of the hypertrees in all distinct ways of forming a cycle. $\mathcal{F}$ has $(s-j)$ hyperedges and $j(b-1)$ components to be arranged into a cycle. The number of $b$-uniform hypercycles having a cycle length $j$ $(2 \leq j \leq s)$ and with $s$ hyperedges is thus

$$\left[ \binom{n}{j(b-1)} j(b-1) \, \frac{\Big((s-j)(b-1)\Big)!}{(s-j)!\,(b-1)!^{s-j}} \right] \left[ \frac{1}{2} \, \frac{\Big(j(b-1)\Big)!}{(b-2)!^{j}} \right], \tag{2}$$

where $n \equiv n(s) = s(b-1)$.

In the above Eq. (2) indeed, the left-hand factor (between brackets) counts the number of forests of rooted $b$-uniform hypertrees, while the right-hand one counts the number of smooth hypercycles labelled with the set $\{1, \ldots, j(b-1)\}$. $j$ distinct hyperedges, and thus labelled with the set $\{1, \ldots, j(b-1)\}$.

Now, $(b-1)!^{-s+j}(b-2)!^{-j} = (b-1)!^{j}(b-1)!^{-s}$ and, since $n = s(b-1)$, we have $(s-j)(b-1) = n - j(b-1)$. So, Eq. (2) simplifies to $\dfrac{n!(b-1)}{2\,(b-1)!^{s}} \, \dfrac{(b-1)!^{j}}{(s-j)!}$, with $j$ ranging from 2 to $s$.

Finally, substituting $n^{s-1}/s^{j}$ for $(b-1)!^{j}$ and summing on $2 \leq j \leq s$, gives the finite sum in Theorem 2: $\displaystyle\sum_{j=2}^{s} \frac{j}{s^{j}(s-j)!} = \frac{1}{s(s-2)!}$, and the result follows. □

## 4 Conclusions and further results

In the above proof of Theorem 2 we are led to distinghish hypercycles according to the lengths of the cycles. Therefore, a question arises: for a given number $n = s(b-1)$ of vertices, what is the cycle length $j$ of the class that contributes most to the number of such hypercycles?

It is shown in [17] that there exists at most $\dfrac{n^{n-2} - f(n,b)}{(b-1)^{(b-2)(n-1)/(b-1)}}$ distinct labelled $b$-uniform hypertrees, where $f(n,b)$ is a lower bound on the number of labelled trees of maximal (vertex) degree exceeding $\Delta = (b-1) + \dfrac{n-1}{b-1} - 2$. In view of extending this result, can we determine a lower bound on the number of labelled trees with no constraint on their maximal (vertex) degree—or, at least, with maximal degree exceeding some $\Delta' < \Delta$? This, for example, by designing some generalized counting techniques based on a bijective or analytic enumeration of $b$-uniform hypertrees. (See also [15, 17]).

In the spirit of [6], some potential applications of Prüfer-like code may also arise as fruitful directions of research.

Encoding algorithms (such as the present one or the algorithm designed in [17]) can be used to generate unique identities (IDs) or PINs. By generating distinct hypertrees with combinatorial enumeration methods, it is possible to compute distinct codes for

each of these hypertrees using such encoding algorithms. All codes generated this way would be distinct and can provide unique IDs. The advantage of the scheme is that no check for repetitions is needed, since IDs generated from distinct hypertrees are unique. Besides, the generation of such codes requires time proportional to the length of the code.

Coding schemes can also be useful for allocating IDs to different users in a system where disjoint sets of users form different groups. Each group is associated with a distinct hypertree, whereas the users within a group are allocated distinct codes of the same hypertree associated with the group. Subgroups can be realized by another level of Prüfer-like encoding. The actual implementation of such group management schemes is an open direction of research (see [17]).

## References


[1] C. Berge, *Graphs and Hypergraphs*, New York, Elsevier, 1973.

[2] S. Caminiti, I. Finocchi, R. Petreschi, On coding labelled trees, *Theoretical Computer Science*, 382:97-108, 2007.

[3] A. Cayley, A Theorem on Trees, *Quart. J. Math. Oxford Series*, 23:376-378, 1889.

[4] H.C. Chen, Y.L. Wang, An efficient algorithm for generating Prüfer codes from labelled trees, *Theory of Computing Systems*, 33:97-105, 2000.

[5] W. Edelson, M.L. Gargano, Modified Prüfer code: $\mathcal{O}(n)$ implementation, *Graph Theory Notes of New York*, 40:37–39, 2001.

[6] F. Flajolet, R. Sedgewick, *Analytic Combinatorics*, Cambridge University Press, 2009.

[7] A. Joyal, Une théorie combinatoire des séries formelles, *Adv. in Math.*, 42:1-82, 1981.

[8] M. Karonski, T. Luczak, The number of sparsely edged uniform hypergraphs, *Discrete Math.*, 171:153-167, 1997.

[9] D.E. Knuth, *The Art of Computer Programing (vol. 1-3)*, 2nd Edition, Addison Wesley, 1973.

[10] G. Labelle, Une nouvelle démonstration combinatoire des formules d'inversion de Lagrange, *Adv. in Math.*, 42:217-247, 1981.

[11] J.W. Moon, Various proofs of Cayley's Formula for Counting Trees, *A Seminar on Graph Theory*, p 0-78, 1967.

[12] A. Nijenhuis, H.S. Wilf, *Combinatorial Algorithms*, Academic Press, New York, 1978.

[13] H. Prüfer, Neuer Beweis eines Satzes über Permutationen, *Arch. Math. Phys.*, 27:742-744, 1918.

[14] V. Ravelomanana, A. L. Rijamamy, Creation and Growth of Components in a Random Hypergraph Process, *Proc. of Cocoon 2006*, LNCS 4112:350-359, 2006.

[15] C. Rényi, A. Rényi, *The Prüfer code for k-trees*, in: Combinatorial Theory and its Applications III, p. 945-971, Erdös, Rényi, Sós (eds.), North-Holland, 1970.

[16] B.I. Selivanov, Perechislenie odnorodnyckh hipergrafov c protstoĭ ciklovoĭ strukturoĭ, *Kombinatornyĭ Analiz*, 2:60-67, 1972.

[17] S. Shannigrahi, S.P. Pal, Efficient Prüfer-Like Coding and Counting Labelled Hypertrees, *Algorithmica*, 54:08-225, 2009.