



Introduction



Généralités



Représentation



Qualité



Optimisation



Bibliographie

# Justification de paragraphe : le Knuth-Plass

~ Exposé GUTenberg, 11 Janvier 2024 ~

Didier Verna

didier@didierverna.net



didierverna.info



@didierverna



didier.verna



in/didierverna

# Introduction

- ▶ T<sub>E</sub>X : standard *de facto* en matière de qualité typographique
- ▶ En particulier : l'algorithme de mise en forme de paragraphe  
*N.b. « paragraphe » vs. « alinéa » hors sujet ici ; cf. [Savary1, 2023] et [Savary2, 2023]*
- ▶ Le « Knuth-Plass »
  - ▶ Développé entre 1977 et 1982
  - ▶ « Probablement l'algorithme le plus intéressant de T<sub>E</sub>X »  
— *Donald Knuth*

**Le Knuth-Plass fait peur...**

# Pourquoi ?

## ▶ Littérature de référence

- ▶ Breaking Paragraphs into Lines [Knuth, 1981] : 66 pages  
*Première mouture, avec formalisation algébrique, exemples et applications*
- ▶ T<sub>E</sub>X : the Program [Knuth, 1986] : 28 pages (+ césure = 58)  
*À jour, entrelacée avec le reste de T<sub>E</sub>X*

## ▶ L'algorithme de Knuth-Plass n'est pas... un algorithme !

- ✓ 1. Un problème de « plus court chemin » *single-pair* dans un graphe orienté acyclique [Dijkstra, 1959] (1956), A\* [Hart, 1968], etc.
- ✓ 2. Une optimisation de type « programmation dynamique » (1950) [Bellman, 1954]
- ✓ 3. Un critère de qualité (fonction de coût) spécifique à T<sub>E</sub>X  
*Compatible avec la programmation dynamique !*
- ✗ 4. Une implémentation (documentée) très impérative, truffée de petites optimisations de très bas niveau

# Au Programme

Généralités : Coupure de Ligne, Tolérance, Passes

Représentation : Graphe des Solutions

Qualité : Pénalités, Démérites, Fonction de Coût

Optimisation : Programmation Dynamique

## Mise en Forme de Paragraphe : Ingrédients

### ▶ Coupure de ligne

✓ ▶ Implicite : espaces inter-mot et points de césure [Liang, 1983]

✗ ▶ Explicite : « pénalités » (`\penalty`)

✗ ▶ Remarque : césures + ligatures = « discrétionnaires »

`\discretionary{-}{ }{ }`, `\discretionary{f-}{fi}{ffi}`

### ▶ Largeur de ligne

✓ ▶ « Glue » élastique inter-mot : dépend de la police utilisée

✗ ▶ Micro-typographie : interlettrage (« tracking »), saillie / crénage de marge (« protrusion »), chasse élastique, *etc.*

### ▶ Remarque : boîtes + glue + pénalités = modèle très expressif

✓ ▶ Justification

✗ ▶ Autres dispositions, saillie rudimentaire, code, *etc.* Cf. [Knuth, 1981]

## Notion de « Tolérance »

- ▶ Complexité théorique exponentielle
  - ▶ Pour  $n$  points de coupures possibles,  $2^n$  solutions
  - ▶ La plupart sont absurdes
- ▶ Élasticité sous contrôle:  $1/3$  *corps de police*  $+1/2 -1/3$ 
  - ▶ Exemple: police 18pt  $\Rightarrow$  6pt, max 9pt ( $6 + 3$ ), min 4pt ( $6 - 2$ )
  - ▶ Réduit considérablement le nombre de solutions *raisonnables*
  - ▶ Critères esthétiques  $\Rightarrow$  *recommandations*
  - ▶ Cependant, limite stricte en compression
- ▶ Tolérance selon T<sub>E</sub>X (« badness »)
  - ▶  $100 * |\% \text{ d'élasticité utilisée}|^3$
  - ▶  $+\infty$  si compression maximale ou étirement  $\geq 465\%$  (10 000)
  - ▶ Dans l'exemple précédent:  $6 + 4.65 * 3 \approx 20\text{pt}$ , soit plus de trois fois l'espacement normal
- ▶ Remarque: T<sub>E</sub>X peut être *infiniment* tolérant...

# Le Knuth-Plass en Trois Passes (max)

## 1. Sans césure, avec `\pretolerance`

- ▶ Plain/ $\text{\LaTeX}$ : 100 par défaut (*i.e.* 100% de l'élasticité disponible)
- ▶ `\pretolerance=-1` pour court-circuiter la passe 1

## 2. Avec césure et `\tolerance`

- ▶ Plain/ $\text{\LaTeX}$ : 200 par défaut (*i.e.* 126% de l'élasticité disponible)
- ▶ Dans l'exemple précédent: 6pt, max 9,78pt (9), min 3,48pt (4)

## 3. Seulement si `\emergencystretch > 0`

- ▶ Toujours avec césure et `\tolerance`

### ▶ Quand ça se passe mal

- ▶  $\text{\TeX}$  produit des lignes débordantes (*overflow box*)
- ▶ `\sloppy` ( $\text{\LaTeX}$ ): `\tolerance=9999 \emergencystretch=3em`
- ▶ Toujours préférer `\emergencystretch` à `\tolerance=10000` ( $+\infty$ )

# Au Programme

Généralités : Coupure de Ligne, Tolérance, Passes

Représentation : Graphe des Solutions

Qualité : Pénalités, Démérites, Fonction de Coût

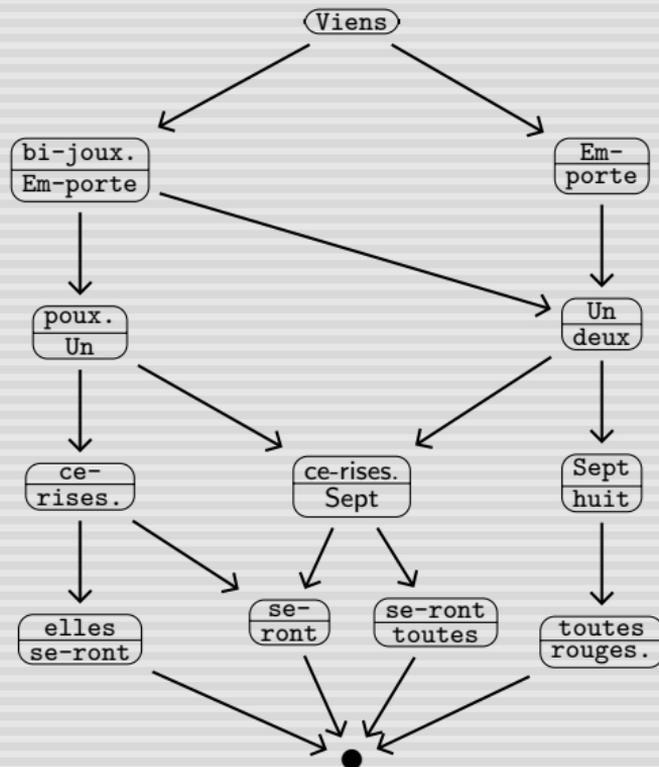
Optimisation : Programmation Dynamique

# Graphe des Solutions

## Exemple

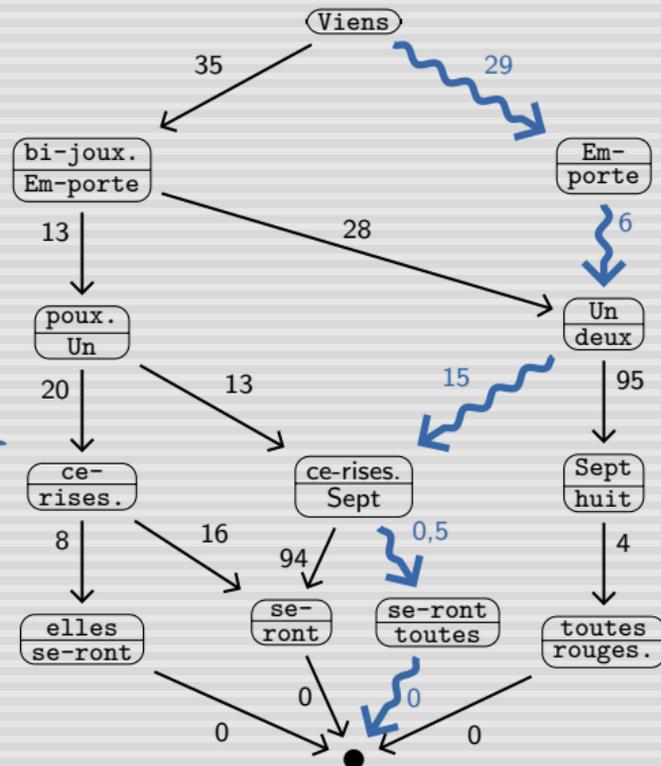
Viens mon chou sur mes ge-noux avec tes jou-joux et tes bi-joux. Em-porte des cailloux pour lan-cer sur les hi-boux pleins de poux. Un deux trois nous irons au bois. Quat' cinq six cueillir des ce-rises. Sept huit neuf dans mon pa-nier neuf. Dix onze douze elles se-ront toutes rouges.

- ▶ Largeur: 288pt (10,12cm)
- ▶ 2<sup>e</sup> passe (`\pretolerance=-1`)
- ▶ Espacement recommandé seulement (`\tolerance=100`)



## Remarques

- ▶ Graphe vs. arbre (partage de noeuds)
  - ▶ Paragraphe rectangulaire
  - ▶ Ou conserver le numéro de ligne
- ▶ Branches mortes / arbre mort
  - ▶ Importance d'une tolérance ajustable
- ▶ Algorithmes gloutons : ligne par ligne
  - ▶ First-Fit, Last-Fit
  - ▶ Best-Fit (mais *define* « best »...)
    - ▶  $\min(\% \text{ élasticité ou badness})$
    - ▶ Nombreux autres choix possibles
- ▶ Avantages
  - ▶ rapides
  - ▶ peu coûteux
- ▶ Inconvénients
  - ▶ Pas de notion de qualité globale
  - ▶ Sensibles aux branches mortes



# Au Programme

Généralités : Coupure de Ligne, Tolérance, Passes

Représentation : Graphe des Solutions

Qualité : Pénalités, Démérites, Fonction de Coût

Optimisation : Programmation Dynamique

# Pénalités Locales

Même localement, l'élasticité n'est pas le seul critère esthétique

## ▶ Quatre types de « pénalités » locales

1. Rappel: `\penalty` (explicite)
2.  $0 \leq \text{\linepenalty}(10)$ : jouer sur le nombre de lignes
3.  $-10\,000 \leq \text{\hyphenpenalty}(50) \leq 10\,000$
4.  $-10\,000 \leq \text{\exhyphenpenalty}(50) \leq 10\,000$ : jouer sur la césure

## ▶ Remarques

- ▶  $\pm 10\,000 \iff \pm \infty$  (coupures interdites ou obligatoires)
- ▶ Rappel: césure  $\implies$  discrétionnaires

ge-noux  $\rightarrow$  "ge" `\discretionary{-}{ }{ }` "noux"  $\rightarrow$  `\hyphenpenalty`

ef-ficace  $\rightarrow$  "e" `\discretionary{f-}{fi}{ffi}` "cace"  $\rightarrow$  `\hyphenpenalty`

plate-bande  $\rightarrow$  "plate-" `\discretionary{}{ }{ }` "bande"  $\rightarrow$  `\exhyphenpenalty`

# Démérites Locaux

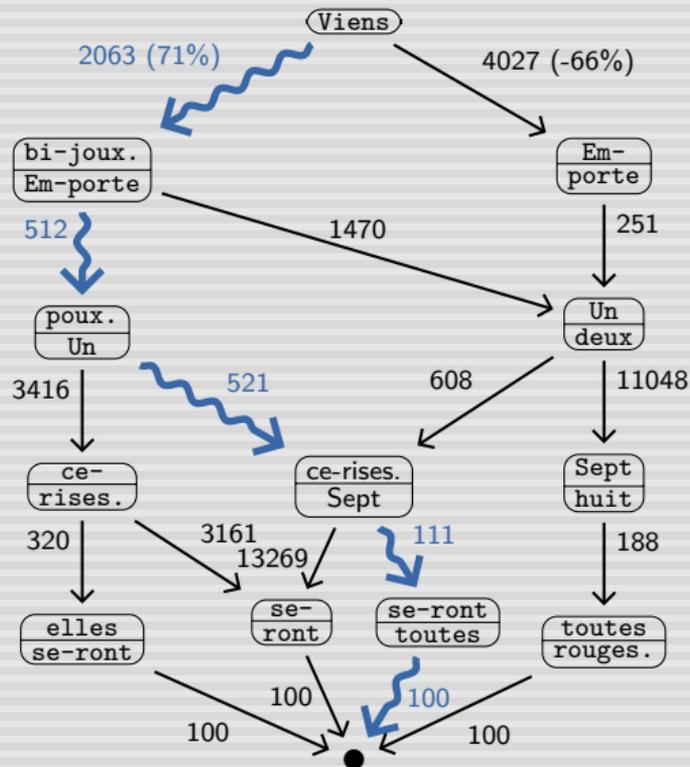
Élasticité (*badness*) + pénalités locales  $\implies$  « démerites » locaux

1. Coupure interdite (ou *badness* > tolérance) : n/a
  2. Coupure pénalisante :  $(\text{pénalité de ligne} + \textit{badness})^2 + \text{pénalité}^2$
  3. Coupure souhaitée :  $(\text{pénalité de ligne} + \textit{badness})^2 - \text{pénalité}^2$
  4. Coupure obligatoire :  $(\text{pénalité de ligne} + \textit{badness})^2$
- Remarques
- Compromis élasticité / césure :  $\sqrt[3]{(\sqrt{10^2 + 50^2} - 10)/100} = 74\%$
  - Pénalité (infinie) non prise en compte en cas de coupure obligatoire

# Application au Best-Fit

►  $\min(\text{démérites locaux})$  

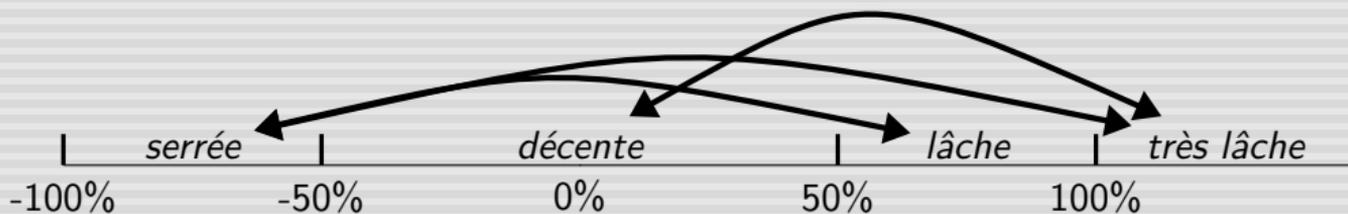
*La césure est plus pénalisante que l'étirement ici (71% < 74%)*



# Démérites Globaux

Ajoutés aux démerites locaux d'une ligne à la suivante

- ▶ Jouer sur la césure
  - ▶  $0 \leq \backslash\text{doublehyphendemerits}$  (10 000): échelles de césure
  - ▶  $0 \leq \backslash\text{finalhyphendemerits}$  (5 000): césure finale (avant-dernière ligne)
- ▶ Jouer sur la différence d'étalement
  - ▶ Quatre classes d'étalement (« *fitness classes* »)
  - ▶  $0 \leq \backslash\text{adjdemerits}$  (10 000): démerites d'adjacence

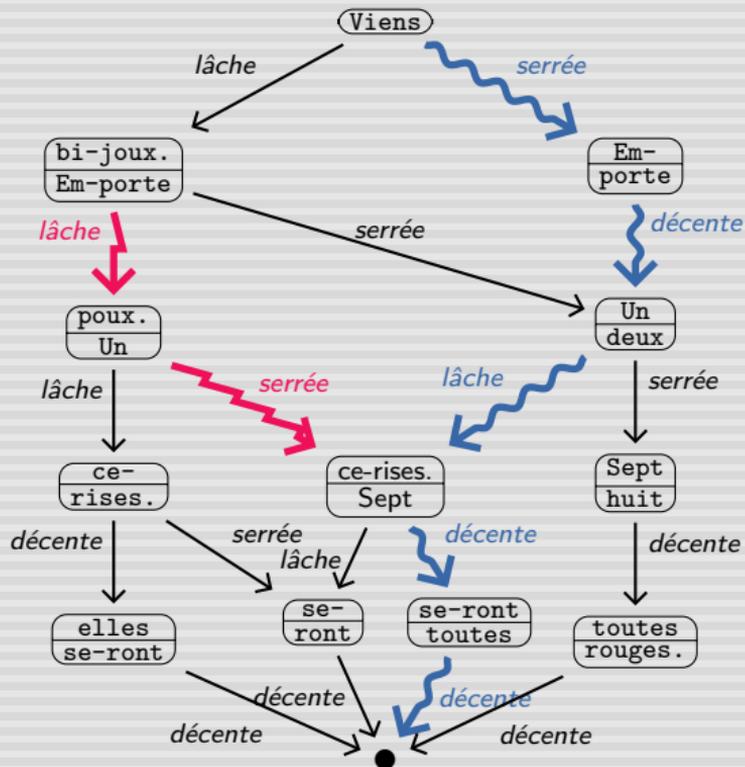


# Application

► Knuth-Plass   
 Note:  $\equiv$  Best-Fit / *badness*

► Chemin problématique 

► Ainsi que plusieurs autres...



## Remarques Finales

- ▶ Démérites
  - ▶ locaux : esthétique horizontale
  - ▶ globaux : esthétique verticale
- ▶ Paramétrage
  - ▶ local (horizontal) en termes de *pénalités*
  - ▶ global (vertical) en termes de *démérites* (pénalité / *badness* au carré)
- ▶ Approche « globale » de  $\text{T}_{\text{E}}\text{X}$   
En réalité, basée uniquement sur des comparaisons de lignes adjacentes...

# Au Programme

Généralités : Coupure de Ligne, Tolérance, Passes

Représentation : Graphe des Solutions

Qualité : Pénalités, Démérites, Fonction de Coût

Optimisation : Programmation Dynamique

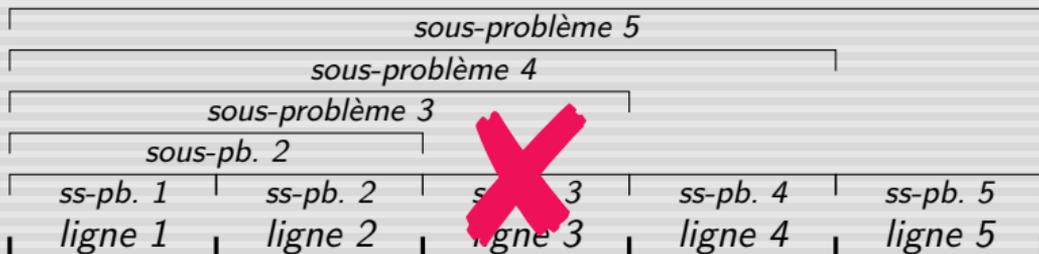
# Contexte

## Rappel : programmation dynamique (1950) [Bellman, 1954]

- ▶ Principe général
  1. Découpage du problème global en (plus petits) sous-problèmes
  2. Résolution des sous-problèmes pour arriver à la solution globale
- ▶ Remarque : « Diviser pour Régner » (*divide and conquer*)
  - ▶ Sous-problèmes indépendants les uns des autres.
  - ▶ Inapplicable ! Justifier la fin d'un paragraphe nécessite d'avoir justifié le début...
- ▶ Caractéristiques
  1. Les sous-problèmes dépendent *séquentiellement* les uns des autres
    - ▶ *Les sous-problèmes sont définis récursivement les uns en fonction des autres*
  2. Le processus de résolution suit nécessairement cette séquence
    - ▶ *La solution globale contient la séquence des sous-solutions*
- ▶ Complexité :  $2^n \implies n^2 \implies nw$

## Qu'appelle-t-on « Sous-Problème » ?

- ▶ Sous-problème  $n =$  s'occuper de la  $n^{\text{e}}$  ligne ? Non !
  - ▶ Le « meilleur » choix pour une ligne ne l'est pas nécessairement pour l'ensemble
  - ▶ Aucune décision définitive, mais un *ensemble* de solutions possibles
- ▶ Sous-problème  $n =$  s'occuper des  $n$  premières lignes
  - ▶ trouver *des* solutions pour la ligne  $n$  en fonction *des* solutions trouvées pour les précédentes

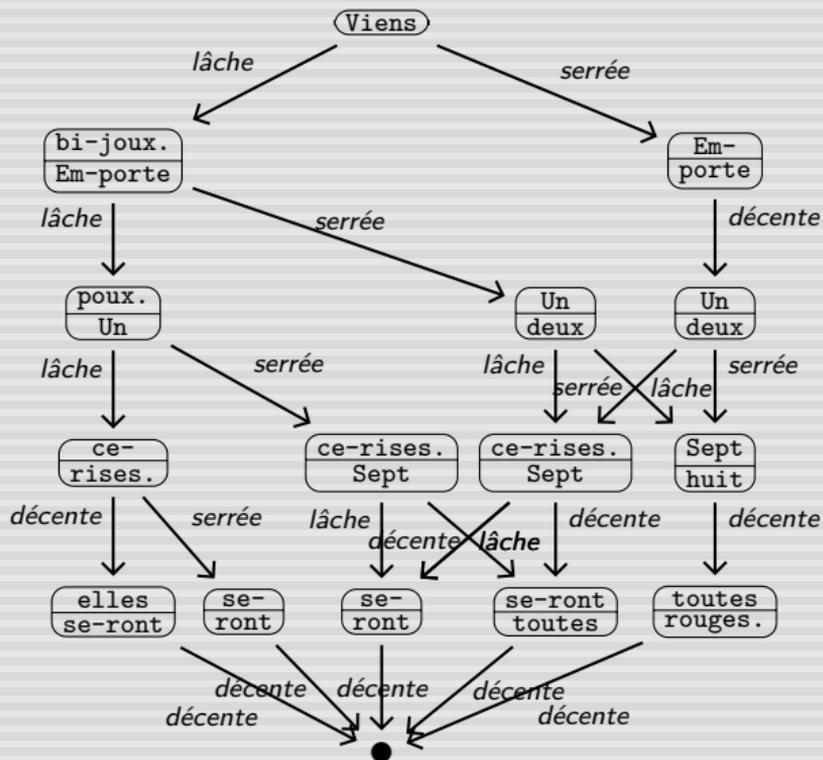


## Conséquences sur le Graphe des Solutions

1. Résolution séquentielle des sous-problèmes
  - ▶ Construction du graphe progressive plutôt que préalable
  - ▶ = *algorithmes gloutons*
2. Aucune décision intermédiaire définitive
  - ▶ Conservation des meilleures branches jusqu'au bout
  - ▶  $\neq$  *algorithmes gloutons*
3. Cependant : suppression des moins bons chemins
  - ▶ « Moins bons » avec certitude absolue !

# Élagage du Graphe

- ▶ Conserver un seul accès, *le meilleur*, à chaque noeud
- ▶ Problème : les classes d'étalement empêchent de définir « meilleur »
- ▶ Solution : des noeuds différents par classe d'étalement entrante
- ▶ Nouveau graphe 🖱️
- ▶ Rappel : problème identique (solution identique) pour des paragraphes non rectangulaires





# Bibliographie

-  [Anéantir Michel \(pour en Finir avec les Livres Moches\)](#)  
Thomas Savary, Blog, 2023.
-  [Le package lua-typo](#)  
Thomas Savary, exposé GUTenberg, 2023.
-  [Breaking Paragraphs into Lines](#)  
Donald E. Knuth, Michael F. Plass.  
Software – Practice and Experience, 11:1119–1184, 1981.
-  [T<sub>E</sub>X: the Program](#)  
Donald E. Knuth.  
Computers and Typsetting, Volume 2, Addison-Wesley, 1986.

## Bibliographie (cont.)

- 🌐 [Word Hy-phen-a-tion by Com-put-er](#)  
Franklin M. Liang.  
Stanford University Ph.D, STAN-CS-83-977, 1983.
- 🌐 [The theory of dynamic programming](#)  
Richard Bellman.  
Bulletin of the American Mathematical Society, 60 (6): 503–516, 1954.
- 🌐 [A Note on Two Problems in Connexion with Graphs](#)  
Edsger W. Dijkstra.  
Numerische Mathematik, 1:269–271, 1959.
- 🌐 [A Formal Basis for the Heuristic Determination of Minimum Cost Paths](#)  
Peter Hart, Nils Nilsson, and Bertram Raphael  
IEEE Transactions on Systems Science and Cybernetics, 4 (2): 100–7, 1968.

## Bibliographie (cont.)



### ETAP

Didier Verna.

[github/didierverna/etap](https://github.com/didierverna/etap).



### Interactive and Real-Time Typesetting for Demonstration and Experimentation: ETAP

Didier Verna.

TUGboat, 44 (2):242–248, 2023.



### ETAP: Experimental Typesetting Algorithms Platform

Didier Verna.

15<sup>th</sup> European Lisp Symposium, pp. 48–52, 2022.



### Justification de Paragraph: le Knuth-Plass

Didier Verna.

Exposé Gutenberg, 11 Janvier 2024.