GreenIT · Fiches DS

Polytech S9 · D. Lopez Pacheco · F. Giroire

Formulaire

Énergie $E = P \times t$ (W × h = Wh ; ÷3600 si t en s)
SCI $(O + M) / R$ · $O = E \times I$
oSCI O / R · **tSCI** $(O + O_{idle} + M + M_{idle}) / R$
ERE (total − réutilisée) / IT ≤ PUE
Bin packing OPT ≥ $\lceil \Sigma s(i) / B \rceil$ · FF, BF ≤ 1,7 OPT
H(s) $= 1 + 1/2 + \ldots + 1/s \le \ln s + 1$ · $H(6) \approx 2{,}45$
Spéculatif tokens/passe = $(1 - \alpha^{\gamma+1}) / (1 - \alpha)$
Poids taille = nb params × octets (FP16 2, INT8 1, INT4 0,5)

Émissions $CO2 = E \times I$ (kWh × g/kWh)
Incorporé $M = TE \times (TiR / EL) \times (RR / ToR)$
PUE total / IT ≥ 1 · **WUE** eau (L) / IT (kWh)
Amdahl gain = $1 / ((1 - p) + p/s)$ · $\max 1/(1 - p)$
FFD ≤ 11/9 OPT + 6/9 · **glouton SC** ≤ H(s) OPT
Cascade coût = $\Sigma \text{Prob}_i \times \text{Cost}_i$
ECE $\Sigma_b (|S_b| / N) \times |\text{acc}(S_b) - \text{conf}(S_b)|$
Score Lab 3 $0{,}1 / (E_{Wh} \times (1 + \Sigma |d_{opt} - d|))$

1 Impact du numérique et mesure

cours D. Lopez (isen-p1)

- **Part du numérique** (2020) : 4 % (Malmodin 2024) à 8-9 % (IEA) de l'énergie · 1,4 % (764 Mt CO2eq) à 4 % des GES · en hausse chaque année
- **Projection** : monde qui baisse pour 1,5 °C, numérique constant à 764 Mt → **3,5 % en 2030, 10 % en 2050** (le dénominateur fond)
- **Du logiciel au physique** : microservices (DeathStarBench) → conteneurs/VM → data centers → réseau → terminaux · 344 GB/mois par foyer (2020)
- **Le matériel s'améliore** (Masanet 2020) **mais la demande dépasse les gains** (IEA 2025) · fabrication et fin de vie non comptées

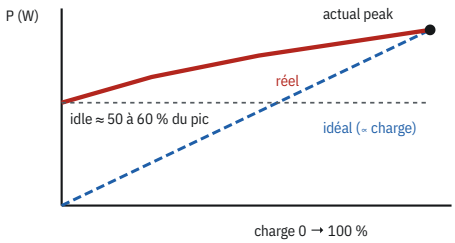
Proportionnalité énergétique

Fan, Weber, Barroso ISCA 2007

Nameplate = Σ max de chaque composant (surestime) · **actual peak** = max mesuré à pleine charge (à utiliser) · **idéal** : 0 W à vide, linéaire

Réseau : conso ∝ nombre de **modules actifs** (ports), pas la bande passante (Mahadevan 2009)

Conséquence : peu de serveurs bien chargés > beaucoup presque vides → modèle ON/OFF, consolidation (ch. 2)

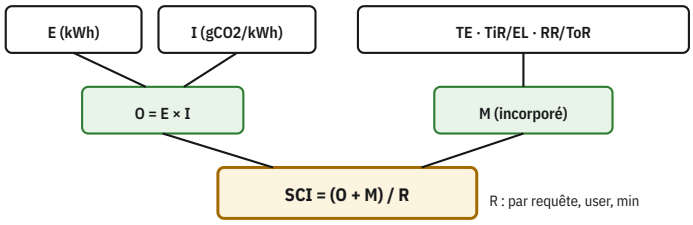


− Limites serveur à 20 % coûte presque autant qu'à 100 % · routeur idem

SCI · Software Carbon Intensity

Green Software Foundation · ISO/IEC 21031:2024 · un taux

Sym.	Sens
E	énergie du logiciel (kWh)
I	intensité carbone de la région (gCO2eq/kWh)
TE	incorporé total (ACV du matériel)
TiR	temps réservé au logiciel
EL	durée de vie attendue
RR / ToR	ressources réservées / totales
R	unité fonctionnelle (requête, user, minute)



Ex. E = 3 kWh, I = 100, M = 60 g, R = 600 req → (300 + 60) / 600 = **0,6 g/req**

+ Forces compare 2 versions à usage égal · inclut le matériel (M) · indépendant de la taille (taux)

− Limites TiR, TE rarement connus · I varie (lieu, heure) · choix de R arbitraire

↔ **Variantes** (Bashir, *Sunk Carbon Fallacy*, SoCC 2024) · **oSCI** = O/R (ignore l'incorporé, déjà émis) · **tSCI** = (tO + tM)/R avec O_{idle} (base au repos) et M_{idle} (incorporé inactif attribué au job)

! Piège SCI n'est pas un total : diviser les émissions par 2 en doublant R ne réduit rien · M dépend de RR/ToR autant que de TE

KPI de data center · PUE, WUE, ERE

The Green Grid

PUE = énergie totale / énergie IT · ≥ 1 · standard 1,6 · mieux 1,1 (slide : « Good = 1.6 » probablement coquille)

WUE = eau sur site / énergie IT (L/kWh) · 3 M L / 1,5 GWh = **2 L/kWh**

ERE = (total − réutilisée) / IT · récompense la chaleur récupérée · sans réutilisation ERE = PUE · ex. 12 total, 6 IT, 3 réutilisés → PUE 2, ERE 1,5

Total site 8 GWh



PUE = 8 / 5 = 1,6 (standard)

à 1,1 : total 5,5 GWh, -2,5 GWh (-31 %)

! Piège PUE mesure le bâtiment, pas l'usage : PUE 1,1 + serveurs vides + électricité charbon = pas vert · ne dit rien de I

Langage, temps, coeurs

Pereira SLE 2017 · Portegies Zwart 2020

Compilés (C, C++, Rust) > interprétés · **pas de gagnant universel** (dépend du problème, CPU + mémoire)

Temps ∝ CO2 (relation linéaire) · plus de coeurs = mieux en général, **trop** = contre-productif · overclocking ↑ conso

↔ **TD** Lab 1 : Python 11,7 s → Java 0,278 s = 42× moins d'énergie à P égal

Mode sombre

mesures Greenspector

AMOLED : pixel noir ≈ éteint · à 100 % de luminosité noir ≪ blanc, bleu · sous 25 % l'écart disparaît

IPS (LCD) : rétroéclairage toujours allumé → toutes couleurs pareilles, seule la **luminosité** compte

! Piège « le sombre économise » : vrai seulement en AMOLED, luminosité haute

Impact de l'IA

IEA 2025 · Luccioni JMLR 2023 · E-waste Monitor 2024 · Giroire « IA et environnement »

Entraînement BLOOM 176 B = **24,7 t CO2eq** (GPU dynamique seul, mix français, sans réseau ni essais) · modèles à refaire plus gros

Inférence : les modèles de raisonnement (o1) déplacent le calcul vers l'inférence (thinking) · requête LLM ≈ **3 Wh** vs recherche 0,3 Wh (×10)

DC 2025 : 500-540 TWh (< 2 % monde, IA 10-12 %) · IA +40 %/an ($1{,}4^5 \approx 5{,}4$) · 2030 : 1250-1500 TWh (5 %, IA > 35 %) · numérique 1400-1600 Mt ≈ 1,5× aviation

E-waste 62 Mt (2022, +82 % vs 2010) → 82 Mt (2030) · IA +1,2 à 5 Mt · **22 %** recyclés · vie 2 à 5 ans

Minerais gallium/indium 70 %, tantale 30-40 %, cobalt 15-20 %, terres rares 15 % · Congo, Birmanie

Eau global 0,01-0,02 % mais local : hyperscale = ville de 10 à 50 k hab. · **mur énergétique** : raccord HT (Virginie, Irlande), SMR

Social travailleurs de la donnée « fantômes » · biais, fake news

! Piège · effet rebond (Jevons 1865) efficacité ↑ → coût ↓ → usage ↑ → conso totale ↑ · 5G 10× plus efficace/bit mais trafic +40 %/an · réponse : sobriété + IA frugale

Exercices corrigés (cours 1)

SCI de deux serveurs (slide 31)

A : TE 400 g, 10 ans, 40 coeurs · B : 50 g, 5 ans, 10 coeurs · job 10 coeurs, TiR = 1 an (hypothèse)

→ M(A) = 400 × 1/10 × 10/40 = **10 g** · M(B) = 50 × 1/5 × 10/10 = **10 g** · égalité : TE 8× plus grand compensé par RR/ToR et EL

PUE et carbone évité

8 GWh total, 5 GWh IT → PUE 1,6 · à 1,1 : 5,5 GWh, −2,5 GWh

→ FR $2{,}5 \cdot 10^6$ kWh × 56 g = **140 t** · DE × 381 g = **952,5 t** · même geste, 6,8× plus en Allemagne

Retrouver 3,5 % et 10 %

764 Mt = 1,4 % → total 2020 ≈ **54,6 Gt** · 3,5 % → 21,8 Gt · 10 % → 7,6 Gt

→ le monde doit baisser de **−86 %** d'ici 2050 : la part explose sans que le numérique croisse

Énergie = puissance × temps

7 W · Python 11,702 s → 7 × 11,702 / 3600 = **22,8 mWh** · Java 0,278 s → **0,54 mWh**

→ ratio 42,1 indépendant de P · limite : P non constante (JIT), d'où CodeCarbon

Même code, quatre intensités (1 kWh)

FR instantané 22 g · FR annuel 56 g · FR 6 janv. 2026 81 g · DE annuel 381 g → ×17,3

→ énergie = propriété du code + machine ; I = lieu + instant · levier : **carbon-aware scheduling**

Pourrait tomber

Pourquoi un serveur non proportionnel pousse à consolider ? · Calculer un SCI avec R au choix et justifier R · ERE < PUE : pourquoi ? · Pourquoi le PUE ne suffit pas à dire qu'un DC est vert ?

2 Algorithmes verts

cours F. Giroire (Inria COATI)

- **Constat** : Internet ≈ 3-4 % et en hausse · infra dimensionnée pour l'heure de pointe, allumée 24/7 · conso quasi constante malgré un trafic cyclique (nuit ↓)
- **3 familles** (combinables) : **re-engineering** (éléments plus sobres, ex. tout optique) · **adaptation dynamique** (DVS : fréquence/tension ; idle logic) · **sleeping/standby** + proxy **NCP** pour garder la présence réseau · **le cours traite la veille**
- **Méthode commune** : modèle ON/OFF → problème combinatoire NP-difficile → heuristique/approximation · VM → bin packing · antennes → set cover · routeurs → EAR

Consolidation de VM · Bin packing

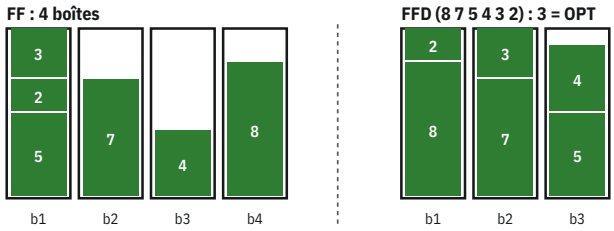
data center · serveurs ON/OFF · fortement NP-complet

Instance items I de taille s(i), capacité B, entier K · **décision** : partition en K parties de somme ≤ B ? · **optim.** K min = OPT(I)

Complexité : pas d'approx. de rapport absolu < 3/2 sauf P = NP · pseudo-poly pour K fixe · poly pour B fixe

```
FF  : 1re boîte ouverte où l'item rentre   ≤ 1,7 OPT
BF  : boîte la plus chargée où il rentre   ≤ 1,7 OPT
FFD : trier décroissant puis FF   ≤ 11/9 OPT + 6/9
```

Approximation : algo poly, solution réalisable, garantie de facteur (2-approx ≤ 2 OPT)



{5, 7, 2, 4, 3, 8}, B = 10 · somme 29 → OPT ≥ 3 · BF : {5,4}{7,2}{3}{8} = 4

Instance (B = 10)	OPT	FF	BF	FFD
{4,8,1,4,2,1}	2	2	2	2
{9,8,2,2,5,4}	4 (le 9 reste seul)	4	4	4
{2,5,4,7,1,3,8}	3	4	4	3
{5,7,3,4,8}	3	4 : {5,3}{7}{4}{8}	3	3
{3,2,4,5,4,2}	2 : {5,3,2}{4,4,2}	·	·	**3** (FFD non optimal)

+ Forces FFD simple, O(n log n), proche de l'optimum en pratique

– Limites aucun glouton optimal · ordre d'arrivée (FF/BF) · VM multi-ressources (CPU + RAM) et migrations non modélisées

! Piège borne → entier inférieur : OPT = 3 → FF ≤ ⌊5,1⌋ = 5, FFD ≤ ⌊39/9⌋ = 4 · borne = pire cas, pas valeur observée

Endormir des antennes · Minimum Set Cover

réseau mobile · coût uniforme 1 · glouton H(s)

U = mobiles à servir · $\mathbf{S_i}$ = mobiles couverts par l'antenne i · min d'ensembles couvrant U · 2e idée : **réassocier** les mobiles pour vider des antennes

```
tant que U non couvert :
  prendre S max |S ∩ non-couverts|
// linéaire en Σ|S_i| (file à seaux)
```

Garantie ≤ H(s) × OPT · H(4) ≈ 2,08 · H(6) ≈ 2,45 · H(100) ≈ 5,19

Ex2 (b)	1	2	3	4	5	6
S1			●	●		
S2	●		●		●	
S3					●	●
S4	●	●				

4, 6, 2 uniques → S1, S3, S4 forcés : **OPT = 3** · glouton prend S2 (3 nouveaux) puis S1, S3, S4 : **4**

↔ **Astuce** élément présent dans un seul ensemble → ensemble forcé · Ex1 (S3, S4 forcés + S1 ou S5) = 3 · Ex2 (a) D forcé, OPT 3, glouton 3

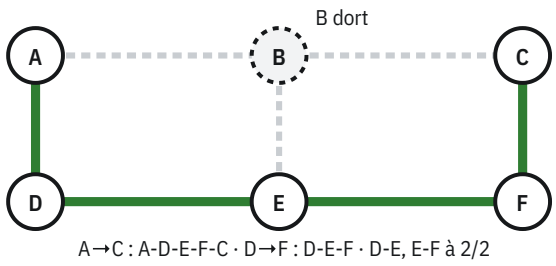
! Piège le glouton n'est jamais < OPT, mais peut être > · Ex3/Ex4 (disques géométriques) : figures à retranscrire du PDF

Energy Aware Routing · LL-SP

réseau coeur · NP-complet, non approximable · MILP ou heuristique

EAR : router toutes les demandes en minimisant les équipements actifs (routeurs, liens) sous capacité · charge ≈ sans effet sur la conso → éteindre interfaces et châssis

```
L = V
tant que L ≠ ∅ :
  lln = noeud le moins chargé de L
  endormir lln
  router tout par plus courts chemins
  si infaisable : rallumer lln, routage précédent
  L = L - {lln}
```



A→C : A-D-E-F-C · D→F : D-E-F · D-E, E-F à 2/2

Liens cap. 2 · demandes A→C, D→F · essais : A (extrémité, non) · **B dort** · C, D, F extrémités · E coupe D→F · 5/6 actifs · + A→F : rien ne dort (heure de pointe)

– Limites heuristique sans garantie · chaque noeud essayé une fois · ordre par charge peut rater mieux · stretch (routes plus longues) et protection aux pannes réduites

↔ **Comparé à** plus court chemin : tous les liens allumés · EAR : des liens éteints, chemins plus longs

Deck annexe « Green Networks »

D. Lopez et F. Giroire 2016

Trafic ×2 / 18 mois · capacité routeur ×2,5 / 18 mois · efficacité silicium (Dennard) ×1,65 → conso routeur ×2,5/1,65 ≈ **1,5 / 18 mois** : le matériel seul ne suffit pas · plan de données 54 % dont 32 % lookup IP + forwarding (Tucker 2009) · ICT 1,9 % des émissions en 2011 · éco vs BAU : 40 TWh, 4 Md$ (2020, telcos UE) · **SDN** contrôle centralisé, tables de règles limitées → compression · **NFV** fonctions virtualisées regroupées · optimisation conjointe réseau + DC · IoT · GeSI SMARTer 2020 : l'ICT économiserait plus qu'elle ne consomme

Questions d'examen (cours 2)

Pourquoi le bin packing pour un DC ?
→ serveur ≈ ON/OFF : coût ∝ nombre de serveurs allumés · items = VM, boîtes = serveurs, B = capacité · min de boîtes = min de serveurs, les autres en veille

Borne basse rapide
→ $\lceil \Sigma / B \rceil$ · {9,8,2,2,5,4} : 30/10 = 3 mais impossible (9 sans 1) → justifier OPT par un argument de remplissage exact

Glouton set cover optimal ?
→ non : contre-exemple Ex2 (b), 4 vs 3 · garantie H(s) seulement

Pourrait tomber
Dérouler FF/BF/FFD sur une nouvelle instance en montrant chaque placement · appliquer LL-SP à un graphe donné en listant charge, essai, résultat · dire pourquoi EAR ne s'approxime pas → MILP

3 IA frugale et inférence distribuée

F. Giroire, Frugal AI parties 1 et 2

- **Continuum Edge-Network-Cloud** : du capteur au cloud, ressources hétérogènes (délai, bande passante, calcul, mémoire, énergie) · edge = latence ↓, transferts ↓, vie privée
- **5 buts** : gros modèle sur petites machines (**MDI**, model splitting) · petits modèles (compression, sélection) · ordonnancer beaucoup de tâches (**DDI**) · adapter le calcul à l'entrée (routing, MoE, cascade, early exit, cache) · combiner (ensemble)

Compression

petits modèles

Pruning (retirer des poids) · **quantization** (FP16 → INT8/INT4) · **distillation** (étudiant imite enseignant) · **low-rank** (factoriser les matrices)

7 B params : FP16 14 Go · INT8 7 Go · INT4 3,5 Go

Small is sufficient

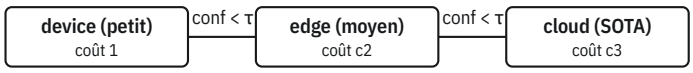
sélection de modèle

Images : meilleur 90 % / 300 M params vs efficace 86 % / 21 M · rendements décroissants (Turgot 1767) · rediriger : **−27,8 %** énergie IA mondiale (31,9 TWh 2025 ≈ 5 réacteurs) pour 7 % d'utilité · Bigger-is-Better : +111 % · slimmable : −70 % énergie pour −2 % précision

Adapter le calcul à l'entrée

routing · MoE · cascade · early exit

Technique	Qui décide	Granularité	Clé
Model routing	routeur entraîné à part, modèles figés (boîtes noires)	modèle entier par requête	RouteLLM, GreenServ (bandit), GPT-5 rapide/raisonnement
MoE	gating appris de bout en bout	experts par token (activation creuse)	capacité découplée du calcul (Shazeer 2017) · Mixtral 8x7B
Cascade	confiance ≥ seuil τ	plusieurs modèles en série	coût = Σ Prob_i Cost_i · device 90 % des cas faciles
Early exit	classifieurs sur couches intermédiaires	un seul réseau, calcul coupé	BranchyNet 2016 · entraînement joint / disjoint / mixte



conf ≥ τ (ex. 0,9) → on répond, sinon on escalade

Ex. petit coût 1, gros coût 10, 20 % escaladés : 1 × 1 + 0,2 × 10 = **3** au lieu de 10

τ ↑ plus d'escalades, coût ↑, précision souvent ↑ · τ ↓ inverse

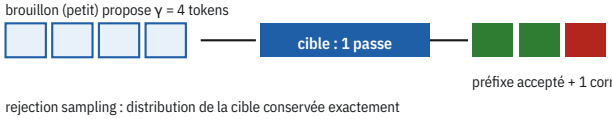
Calibration : réseaux sur-confiants (Guo 2017) · temperature scaling p = softmax(z/T), T appris sur validation, poids figés

ECE : 60 préd. conf 0,9 acc 0,8 + 40 préd. conf 0,6 acc 0,6 → 0,6 × 0,1 + 0,4 × 0 = **0,06**

! Piège cascade ≠ early exit (plusieurs modèles vs un seul coupé) · routing ≠ MoE (après coup sur modèles figés vs appris dans un seul réseau) · une confiance mal calibrée fausse tout le compromis

Décodage spéculatif et caches

LLM · decode séquentiel, limité par la mémoire



rejection sampling : distribution de la cible conservée exactement

α proximité brouillon/cible · **c** coût relatif du brouillon · **γ** tokens proposés

Leviathan 2023 : α = 0,8, γ = 4 → $(1 - 0{,}8^5)/0{,}2 \approx$ **3,36 tokens/passe** · c = 0,1 → gain ≈ 3,36 / (γc + 1) ≈ 2,4×

Self-speculative (Draft & Verify, LayerSkip, SWIFT) : même LLM, brouillon en sautant des couches

KV cache (serveur, dans une requête) : K, V des tokens vus gardés · mémoire ∝ contexte

Prompt caching (entre requêtes) : préfixe identique réutilisé · correspondance **exacte** : 1 caractère changé en amont invalide la suite

Cache sémantique (client) : embedding + similarité > seuil → réponse en cache, 100 % d'économie sur succès · risques : seuil, réponses périmées, fuite entre users (MeanCache)

– Limites brouillon = params, mémoire, déploiement en plus · **moins de temps ≠ toujours moins d'énergie** (Dutta 2026)

4 Lab 1 · Profiling (perf, flame graph, Amdahl)

perf et la démarche

best_offer.py : 5000 hôtels × 1000 recherches

```
PYTHONPERFSUPPORT=1 perf record -g \
  -o ~/perf.data python3 best_offer.py
perf script -i ~/perf.data > stack.txt
# speedscope, mode Left Heavy
java -XX:+PreserveFramePointer \
  -XX:+DumpPerfMapAtExit BestOffer
```

Échantillonnage de la pile (pas de trace de chaque appel) · **-g** = toute la chaîne d'appel · **self** = code propre · **children** = self + appelées · flame graph : largeur = temps cumulé, hauteur = profondeur

Docker : --privileged, SYS_ADMIN, perf_event_paranoid=-1

Version	Changement	Temps
v0	sorth tri insertion O(n²) 48 %	11,70 s
v1	sorted() O(n log n)	6,00 s
v2	services : sets O(n+m)	4,49 s
v3	**architecture** : filtrer distance avant score (748/5000)	2,10 s
Java	port v3, JIT	0,278 s (×7,5)

Hotspot qui se déplace : sorth → calc_svc_score (55 %) → haversine (12 → 63 % sans ralentir)

Amdahl gain = 1 / ((1 − p) + p/s) · plafond 1/(1 − p) · fonction à 5 % → gain max 1,05

v0 → v1 : p = 0,48, s → ∞ : 1/0,52 = **1,92** prédit · mesuré 11,70/6,00 = 1,95

Java : types statiques, JIT natif, même nb d'appels, coût/appel ↓ · run court : > 50 % des échantillons dans le JIT (C2Compiler) · **Rust** AOT : pas de warmup, JIT ni GC · inlining : tout fond dans main

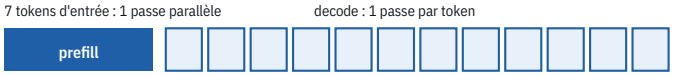
+ Leçon profiler avant d'optimiser · re-profiler après chaque changement · algo/archi > langage seul, puis langage en plus

! Piège trace Rust x86_64 vs nôtre arm64 : comparer la structure, jamais les temps · set dédoublonne (nuance v2)

5 Lab 2 · Énergie d'une requête LLM (Gemini, EcoLogits)

Tokens, prefill, decode, thinking

gemini-3-flash-preview · EcoLogits 0.11



173 tokens de sortie (158 thinking + 15 visibles) → 1 + 173 = 174 passes

prix : in 0,50 $/M, out 3,00 $/M (thinking inclus)

```
EcoLogits.init(providers=["google_genai"])
client = genai.Client(...)   # APRÈS init
# streaming : impacts sur le DERNIER chunk
chunk.impacts.energy.value.min / .max
```

Thinking compte dans max_output_tokens et est facturé : 100 → blague tronquée · 512 OK · 158/173 = 91 % de la sortie

EcoLogits estime (modèle : taille supposée, tokens, infra), ne mesure pas · intervalle **0,000335 à 0,000678 kWh** · warnings normaux : architecture non publiée, multimodal

Prix $7 \times 0{,}50/10^6 + 173 \times 3/10^6$ = **0,000522 $** · sortie 99 % · $\times 10^6$ req ≈ 522 $ · sans thinking sortie ÷11

CO2 0,000335 kWh × 56 g ≈ **19 mg** (FR) · 0,000678 × 381 ≈ 258 mg (DE) · 10^6 req × 0,335 Wh = 335 kWh ≈ 18,8 kg FR

↔ **Profils** entrée courte / sortie longue (génération, nouvelle 2000 mots) = cher · entrée longue / sortie courte (classification, oui/non sur contrat) = moins cher à volume égal

! Piège secrets : .env + --env-file, jamais en dur ni dans l'image · verify=False toléré seulement (certificat auto-signé du labo)

6 Lab 3 · CodeCarbon, distance OSA, spellchecker

CodeCarbon : puissance → énergie → CO2

codecarbon 3.3.1 · Apple M4 Pro

```
codecarbon detect
codecarbon monitor --offline \
  --country-iso-code FRA -- python3 prog.py
# emissions.csv : energy_consumed (kWh)
#                 emissions (kgCO2eq)
force_carbon_intensity_g_co2e_kwh = 81
```

Cas	I (g/kWh)
FR annuel (offline)	56
DE annuel (offline)	381 (×6,7)
FR instantané (online)	22
FR 6 janv. 2026 (froid, gaz)	81

Offline : intensité moyenne **annuelle** du pays · **online** : instantanée (Electricity Maps, token + experiment_id)

wdistance : même énergie FRA/DEU ($4{,}5 \cdot 10^{-8}$ kWh), émissions ×6,7

! Piège token Electricity Maps resté dans ~/.codecarbon.config → --country-iso-code ignoré (FRA = DEU = 22) · offline = [codecarbon] nu

Distance Damerau-Levenshtein (OSA)

DP O(m × n) · symétrique

```
d[i][j] = min(d[i-1][j] + 1,       # suppr.
              d[i][j-1] + 1,       # insert.
              d[i-1][j-1] + cost)  # subst.
si a[i-1]=b[j-2] et a[i-2]=b[j-1] :
  d[i][j] = min(d[i][j], d[i-2][j-2] + 1)
```

hievr/hiver = **1** (Levenshtein : 2) · hiv/hiver 2 · ver/hiver 2 · vre/hiver 3

		ε	h	i	v	e	r
		0	1	2	3	4	5
ε		1	0	1	2	3	4
h		2	1	0	1	2	3
i		3	2	1	1	1	2
e		4	3	2	1	1	2
v		5	4	3	2	2	1
r		6	5	4	3	3	2

case (v, e) : transposition « ev » / « ve » depuis d[2][2] = 0 → 1

Spellchecker : rendre la mesure possible

235 976 mots · Tom Sawyer : 3136 inconnus

Version	Idée	Résultat
naïf	chaque inconnu × tout le dico ≈ $7{,}4 \cdot 10^8$ distances	> 10 min, non fini
challenge 1	buckets par longueur L−2..L+2 (d ≤ 2 ⇒ |Δlong| ≤ 2) + DP plafonnée (arrêt si ligne > 2)	> 5 min
challenge 2	mêmes buckets + rapidfuzz (C), score_cutoff=2	**90 s, 0,251 Wh, 14 mg**

↔ **Score** 0,1 / (0,251 × (1 + 0)) ≈ **0,40** (distances exactes) · dico en set : appartenance O(1)

7 Parallèles cours ↔ TD et synthèse

Notion du cours	Où on l'a manipulée	Ce qu'il faut savoir écrire
E = P × t, temps ∝ CO2	Lab 1 (11,7 → 0,278 s)	diviser le temps par k divise l'énergie par k, à P constante
O = E × I du SCI	Lab 3 (FRA/DEU, online/offline)	même énergie, émissions selon lieu et instant
Coût de l'inférence IA	Lab 2 (thinking 91 %)	decode séquentiel, la sortie domine énergie et prix
Complexité algorithmique	Lab 1 ($O(n^2)$ → n log n), Lab 3 (buckets)	l'algo/architecture bat le langage, puis le langage s'ajoute
Langages compilés plus verts	Lab 1 (Java JIT, Rust AOT), Lab 3 (rapidfuzz C)	pas universel ; JIT a un coût de démarrage
Estimer vs mesurer	EcoLogits (estime), CodeCarbon (mesure + modèle), perf (profile)	intervalle min/max, répéter les runs (10 au Lab 3)
Non-proportionnalité → veille	Cours 2 : bin packing, set cover, LL-SP	ON/OFF ⇒ minimiser le nombre d'équipements allumés
Small is sufficient, cascade	Lab 2 (choisir un modèle flash)	petit modèle d'abord, escalader si confiance faible

Mesurer proprement

→ répéter (10 runs), moyenne + dispersion, séries qui ne se chevauchent pas · même machine, données, config · profil qui change là où on a corrigé

Leviers de réduction, du code au système

→ algo (complexité) · archi (ne pas calculer l'inutile) · langage · matériel bien chargé (consolidation) · lieu/heure (I) · modèle plus petit (frugal) · sobriété (rebond)

Question ouverte type

« L'IA est-elle une innovation verte ? » → pour : smart grids, matériaux (électrolyte −70 % lithium, 32 M → 18 candidats en 80 h), transports · contre : rebond, DC +40 %/an, eau locale, minerais, e-waste · conclure : efficacité nécessaire mais pas suffisante, sobriété + mesure (SCI)

Piège général

Une meilleure efficacité (PUE, J/bit, Wh/requête) ne garantit pas une baisse du total · toujours distinguer **énergie** (kWh), **émissions** (gCO2eq), **taux** (SCI par R)

cours.lsro.fr/greenit · sources : slides isen-p1, Green Algorithm Design, Frugal AI 1 et 2, IA et environnement, sujets et comptes rendus des Labs 1 à 3. Valeurs des labs = mesures réelles du compte rendu.