

Sélection à contribution optimale exacte à l'échelle génomique : un solveur « support-first », sans matrice, validé sur panels de marqueurs réels

Adel Kaleche

ORCID [0009-0005-7566-1422](https://orcid.org/0009-0005-7566-1422)

Chercheur indépendant, France.

Version française ; le manuscrit de référence est en anglais.

11 octobre 2026

Résumé

La sélection à contribution optimale (OCS) arbitre entre gain génétique et borne quadratique de coancestrie. La parenté dense $n \times n$ coûte $O(n^2m)$ à construire et $O(n^2)$ à stocker. Nous présentons **support-first**, un solveur à ensemble actif utilisant de petits supports de travail, une forme fermée par face et des certificats KKT pour l'optimum convexe continu. Deux mesures sont distinctes. Recevant $\mathbf{G}$ dense, un prototype NumPy est environ 12 à 133× plus rapide qu'optiSel sur les panels testés à des cibles approximativement appariées du proxy $\Delta F_{\text{proxy}} = 1\%$ ($N_e = 1/\sum c_i^2 = 50$). La voie Rust sans matrice livrée coûte $O(nm)$ par produit ; elle est environ 3× plus rapide sur le blé, 6× sur la souris, mais 0,3× aussi rapide sur le porc à nombreux marqueurs. Elle résout l'instance structurée $n = 40000$, $m = 1000$ en 0,646 s. La parenté dense correspondante occuperait 11,9 Gio contre 0,30 Gio pour les génotypes denses : cette réduction n'est pas un échec mémoire mesuré d'un solveur dense. Des points d'entrée Rust packés stockent les génotypes entiers à deux bits ; le stockage théorique 500000×50000 occuperait environ 6,25 Go, contre deux téraoctets de parenté dense. Les bindings R/Python prennent actuellement des génotypes denses. Des solveurs indépendants étayent l'accord numérique ; les plafonds individuels sont gérés. Les supports observés restent modestes aux points testés, sans garantie universelle avec ridge ou plafonds finis arbitraires. Les étiquettes ΔF_{proxy} désignent un proxy de contributions, pas une consanguinité génomique réalisée mesurée. Support-first offre une formulation OCS continue exacte avec un compromis vitesse-mémoire documenté.

1 Introduction

L'amélioration génétique du bétail et des plantes doit tenir deux objectifs en tension : maximiser le mérite génétique de la génération suivante, et conserver la diversité génétique dont l'érosion entraîne la dépression de consanguinité et ferme la porte à la réponse future à la sélection. La sélection à contribution optimale (OCS ; Meuwissen 1997) explicite ce compromis. Étant donné la valeur génétique estimée de chaque candidat et la matrice de parenté entre candidats, l'OCS choisit les contributions génétiques proportionnelles $\mathbf{c}$ qui maximisent le gain espéré $\mathbf{b}^\top \mathbf{c}$ sous une borne sur le score quadratique de parenté, $\mathbf{c}^\top \mathbf{G} \mathbf{c} \leq k$, les contributions étant non négatives et de somme un — et, dans un schéma biparental distinguant les sexes, réparties de sorte que pères et mères fournissent chacun la moitié. L'OCS est le cadre de fait pour gérer la diversité, tant en sélection commerciale qu'en programmes de conservation, et la matrice de parenté génomique (VanRaden 2008) a largement remplacé son prédécesseur fondé sur le pedigree comme $\mathbf{G}$ qu'elle contraint.

Avec des données génomiques, il s’agit d’un programme quadratique convexe à contrainte quadratique, équivalent à un programme conique du second ordre. Construction de parenté dense et factorisation conique peuvent devenir coûteuses quand le nombre de candidats croît. La matrice dense coûte $O(n^2)$ en mémoire et $O(n^2m)$ à construire ; la limite effective dépend du matériel et des autres allocations. Les supports modestes observés ici motivent une méthode spécialisée, sans supposer une parcimonie universelle ni l’incapacité des solveurs généralistes à exploiter la structure.

Les méthodes OCS utilisent plusieurs représentations et optimiseurs : conditions lagrangiennes pour Meuwissen (1997) et Gencont2 (Dagnachew & Meuwissen 2016), formulation conique à point intérieur pour optiSel (Wellmann 2019), approches semi-définies (Pong-Wong & Woolliams 2007) ou ADMM/JuMP (Waldmann 2025). AlphaMate (Gorjanc & Hickey 2018) utilise l’évolution différentielle et alloue aussi les accouplements discrets. Yamashita, Mullin & Safarina (2018) exploitent l’inverse creux du pedigree : stocker la parenté dense n’est pas intrinsèque à tout solveur OCS. Les références génomiques denses évaluées ici matérialisent $\mathbf{G}$; notre contribution combine support croissant, produits sans matrice depuis les génotypes et résolutions restreintes en forme fermée.

Nous présentons **support-first** (« le support d’abord »), un solveur OCS exact bâti sur ces deux faits. Il maintient un petit support de travail, initialisé avec le meilleur candidat de chaque sexe et étendu par une règle d’ensemble actif fondée sur le coût réduit : les candidats sont évalués contre les multiplicateurs courants, le plus avantageux est introduit, ceux qui deviennent négatifs sont retirés, le tout vers une unique borne fixée k du score quadratique de parenté. Chaque sous-problème à support fixé — maximiser une forme linéaire sur un ellipsoïde intersecté avec les contraintes affines de somme (et de sexe) — est résolu en forme fermée, en éliminant les contraintes d’égalité et en se ramenant à une équation quadratique scalaire dans le multiplicateur actif, sans solveur itératif interne. Les produits de parenté $\mathbf{Gc}$ sont formés sans matrice à partir de la matrice de génotypes centrée $\mathbf{Z}$ sous la forme $\varepsilon \mathbf{c} + \mathbf{Z}(\mathbf{Z}^\top \mathbf{c})/s$, de sorte que le solveur ne construit ni ne stocke jamais la matrice dense $n \times n$ $\mathbf{G}$.

Les ingrédients pris isolément sont classiques, et nous n’en revendiquons aucun. Les méthodes d’ensemble actif qui suivent un petit ensemble de travail d’actifs détenus descendent de l’algorithme de la ligne critique de Markowitz (1956) pour la sélection moyenne-variance long-only, dont l’OCS est l’analogue contraint en parenté ; la forme fermée pour un objectif linéaire sur un ellipsoïde intersecté avec des contraintes linéaires est un résultat de valeur propre contrainte / équation séculaire (Gander, Golub & von Matt 1989) ; et le produit sans matrice $\mathbf{Z}(\mathbf{Z}^\top \mathbf{c})$ est standard en prédiction génomique (VanRaden 2008 ; Legarra & Misztal 2008). Notre contribution est leur **synthèse, spécialisée à l’OCS génomique** : une génération de colonnes fondée sur le coût réduit qui fait croître un support restreint vers une *unique* borne du score quadratique de parenté — par opposition au balayage paramétrique de *toute* la frontière efficace de l’algorithme de la ligne critique — avec le produit de parenté évalué sans matrice, de sorte que le coût d’une résolution suit la taille du support et le nombre de marqueurs, plutôt que la matrice dense $n \times n$. Les méthodes OCS antérieures touchent à la parcimonie du support, mais par l’autre bout : la méthode de Meuwissen (1997) résout sur l’ensemble complet des candidats et *élague* ceux dont la contribution devient négative — un ensemble actif décroissant sur le système dense $n \times n$. La nôtre fait croître le support par le bas via une génération de colonnes à coût réduit et ne forme jamais ce système ; à notre connaissance aucune méthode OCS antérieure ne combine ainsi un ensemble actif croissant et sans matrice fondé sur les génotypes ; le bénéfice tient à la combinaison plutôt qu’à l’une quelconque de ses parties.

Nous évaluons trois panels publics : blé CIMMYT ($n = 599$), porc PIC ($n = 3534$, environ 53k SNP) et souris en population hétérogène ($n = 1814$, sexe enregistré), avec un sous-panel porcin sexué. Les contre-vérifications coniques synthétiques et les comparaisons à coancestrie réalisée appariée évaluent l’optimalité numérique. Les comparaisons opérationnelles historiques documentent aussi des utilisations différentes de la borne demandée ; leur table de temps ne prouve pas des objectifs identiques à chaque borne.

Deux avantages se lisent séparément. Recevant $\mathbf{G}$, le prototype NumPy est 2 à 482× plus rapide qu’optiSel aux points historiques testés (environ 12 à 133× à $\Delta F_{\text{proxy}} = 1\%$). Ces mesures concernent ce prototype, pas un point d’entrée Rust dense livré. La voie Rust sans matrice évite de stocker $\mathbf{G}$ à $O(nm)$ par produit. Face à la formulation dense Clarabel en Rust, la table synthétique à borne plus serrée donne 716× à $n = 10000$; le facteur historique d’environ 76000× appartient à une borne différente, lâche. AlphaMate est comparé sur la seule relaxation continue commune à coancestrie appariée.

L’expérience de passage à l’échelle démontre l’avantage de stockage. Au proxy de contributions fixé $\Delta F_{\text{proxy}} = 1\%$, les populations structurées ont des supports d’environ 80–103 quand n passe de 1000 à 40000. Le run publié $n = 40000$, $m = 1000$ a pris 0,646 s (environ 0,65 s). Une $\mathbf{G}$ dense occuperait 11,9 Gio, environ 40× les 0,30 Gio de $\mathbf{Z}$ dense. La construction dense était omise au-delà de $n = 30000$ par politique du benchmark; ce n’est ni une erreur mémoire observée ni la preuve que tout portable ou concurrent échoue.

2 Méthodes

2.1 Sélection à contribution optimale

Pour n candidats à la sélection de valeurs génétiques estimées $\mathbf{b} \in \mathbb{R}^n$ et de matrice de parenté génomique $\mathbf{G} \in \mathbb{R}^{n \times n}$, la sélection à contribution optimale choisit les contributions génétiques proportionnelles $\mathbf{c}$ qui résolvent

$$\text{maximiser } \mathbf{b}^\top \mathbf{c} \quad \text{sous} \quad \mathbf{A}\mathbf{c} = \mathbf{d}, \quad 0 \leq \mathbf{c} \leq \mathbf{u}, \quad \mathbf{c}^\top \mathbf{G}\mathbf{c} \leq k.$$

Les contraintes affines $\mathbf{A}\mathbf{c} = \mathbf{d}$ encodent le budget de contribution. Dans la forme simplexe, une seule ligne $\mathbf{A} = \mathbf{1}^\top$, $\mathbf{d} = 1$, impose $\sum_i c_i = 1$. Dans la forme *sexuée* pour un schéma biparental distinguant les sexes, $\mathbf{A}$ est la matrice d’incidence de sexe $2 \times n$ (ligne 1 l’indicatrice mâle, ligne 2 l’indicatrice femelle) et $\mathbf{d} = (\frac{1}{2}, \frac{1}{2})^\top$, imposant $\sum_{\text{mâles}} c_i = \sum_{\text{femelles}} c_i = \frac{1}{2}$. Le plafond par candidat $\mathbf{u}$ borne la contribution d’un individu donné — un sélectionneur laisse rarement un seul parent dominer une génération — et est inactif à $\mathbf{u} = 1$. Le solveur borne le score quadratique de parenté $q_G = \mathbf{c}^\top \mathbf{G}\mathbf{c} \leq k$. Avec la convention de parenté additive utilisée ici, la matrice de coancestrie est $\mathbf{K} = \mathbf{G}/2$, donc la coancestrie de groupe est $q_K = \mathbf{c}^\top \mathbf{K}\mathbf{c} = q_G/2$, sous la borne $k/2$. Ce score de groupe ne spécifie pas la consanguinité réalisée de la descendance, qui dépend aussi des accouplements et de la population de référence. Nous indexons les points de fonctionnement par le nombre effectif de parents contributeurs $N_e = 1/\sum_i c_i^2$, avec le proxy de contributions $\Delta F_{\text{proxy}} = 1/(2N_e)$ (Woolliams & Bijma 2000), et choisissons la borne k atteignant un N_e cible par bisection — une ligne « $\Delta F_{\text{proxy}} = 1\%$ » est donc la borne donnant $N_e = 50$. Ici N_e est le ratio de participation des contributions et ΔF_{proxy} un proxy de leur concentration. Aucun des deux ne mesure la consanguinité génomique réalisée ni n’estime la taille efficace démographique; la recommandation FAO sert de contexte, pas d’équivalence validant ces étiquettes. Nous prenons $\mathbf{G}$ comme la matrice de parenté génomique de VanRaden, $\mathbf{G} = \mathbf{Z}\mathbf{Z}^\top/s + \varepsilon\mathbf{I}$, où $\mathbf{Z}$ est la matrice $n \times m$ des génotypes centrés par deux fois les fréquences alléliques, $s = 2\sum_j p_j(1 - p_j)$, et ε une faible régularisation ridge assurant la définie-positivité (la même matrice régularisée pour les contrôles de précision; les colonnes historiques de temps calibraient leurs bornes séparément). Le programme est un programme quadratique convexe à contrainte quadratique, équivalent à un programme conique du second ordre.

2.2 Observation du support et structure du problème restreint

Support-first utilise une observation empirique et une propriété algébrique. Le support total $S = \{i : c_i^* > 0\}$ est modeste dans les instances testées. Avec plafonds, il comprend les contributions

libres $F = \{i : 0 < c_i^* < u_i\}$ et celles au plafond positif $U_+ = \{i : c_i^* = u_i > 0\}$, donc $|S| = |F| + |U_+|$. Sur une face fixée, la résolution des coordonnées libres a une forme fermée dans le cas non dégénéré à parenté active. L'enjeu est d'identifier cette face ; cela ne prouve pas une borne de pire cas du support.

2.3 Forme fermée sur un support fixé

Fixons un support S de taille $|S|$, et notons $\mathbf{G}_S$, $\mathbf{b}_S$, $\mathbf{A}_S$ les restrictions à S ($\mathbf{A}_S$ est $q \times |S|$ avec $q = 1$ ou 2 lignes d'égalité). Pour le problème restreint non plafonné avec $\mathbf{A}_S$ de plein rang en lignes, objectif non constant sur son enveloppe affine et parenté active avec $\lambda > 0$, la stationnarité s'écrit

$$\mathbf{b}_S = \mathbf{A}_S^\top \mu + 2\lambda \mathbf{G}_S \mathbf{c}_S, \quad \mathbf{A}_S \mathbf{c}_S = \mathbf{d}, \quad \mathbf{c}_S^\top \mathbf{G}_S \mathbf{c}_S = k,$$

avec multiplicateurs $\mu \in \mathbb{R}^q$ et $\lambda > 0$. Résoudre la stationnarité pour $\mathbf{c}_S$ donne $\mathbf{c}_S = \frac{1}{2\lambda} \mathbf{G}_S^{-1} (\mathbf{b}_S - \mathbf{A}_S^\top \mu)$. Imposer les contraintes affines élimine μ : avec la matrice $q \times q$ $\mathbf{P} = \mathbf{A}_S \mathbf{G}_S^{-1} \mathbf{A}_S^\top$ et le vecteur $\mathbf{q}_v = \mathbf{A}_S \mathbf{G}_S^{-1} \mathbf{b}_S$,

$$\mu = \mathbf{P}^{-1} (\mathbf{q}_v - 2\lambda \mathbf{d}), \quad \text{d'où} \quad \mathbf{c}_S = \mathbf{g} / (2\lambda) + \mathbf{h},$$

où $\mathbf{g} = \mathbf{G}_S^{-1} (\mathbf{b}_S - \mathbf{A}_S^\top \mathbf{P}^{-1} \mathbf{q}_v)$ et $\mathbf{h} = \mathbf{G}_S^{-1} \mathbf{A}_S^\top \mathbf{P}^{-1} \mathbf{d}$ vérifient $\mathbf{A}_S \mathbf{g} = \mathbf{0}$ et $\mathbf{A}_S \mathbf{h} = \mathbf{d}$, de sorte que $\mathbf{A}_S \mathbf{c}_S = \mathbf{d}$ pour tout λ . En substituant dans l'ellipsoïde actif $\mathbf{c}_S^\top \mathbf{G}_S \mathbf{c}_S = k$ et en posant $\alpha = \mathbf{g}^\top \mathbf{G}_S \mathbf{g}$, $\beta = \mathbf{g}^\top \mathbf{G}_S \mathbf{h}$, $\gamma = \mathbf{h}^\top \mathbf{G}_S \mathbf{h}$, on obtient une unique équation quadratique scalaire dans le multiplicateur :

$$4(\gamma - k) \lambda^2 + 4\beta \lambda + \alpha = 0.$$

On prend la racine positive maximisant $\mathbf{b}_S^\top \mathbf{c}_S$. Comme $\mathbf{G}_S \mathbf{g} = \mathbf{b}_S - \mathbf{A}_S^\top \mathbf{P}^{-1} \mathbf{q}_v$ et $\mathbf{G}_S \mathbf{h} = \mathbf{A}_S^\top \mathbf{P}^{-1} \mathbf{d}$ sont déjà disponibles, α, β, γ ne coûtent que des produits scalaires — pas de seconde factorisation. Une résolution restreinte est donc : une factorisation de Cholesky de $\mathbf{G}_S$, des remontées pour les $q + 1$ seconds membres $[\mathbf{A}_S^\top \mid \mathbf{b}_S]$, une inversion $q \times q$ ($q \leq 2$) et une équation quadratique scalaire — sans itération interne. Le cas simplexe ($q = 1$, $\mathbf{A}_S = \mathbf{1}^\top$) est identique et se réduit à une équation quadratique scalaire en μ . Quand $|S| = q$ et $\text{rang}(\mathbf{A}_S) = q$, les égalités déterminent $\mathbf{c}_S$ — $(\frac{1}{2}, \frac{1}{2})$ pour un candidat de chaque sexe — et la faisabilité se vérifie directement, sans division par λ . Des lignes de budget dépendantes ou un objectif constant sur l'enveloppe affine doivent être traités séparément. La résolution renvoie « infaisable sur S » quand S manque d'un sexe ou que l'ellipsoïde ne rencontre pas l'enveloppe affine, signal pour agrandir S .

2.4 L'algorithme support-first

Le support est trouvé par génération de colonnes en ensemble actif. Il est initialisé avec le meilleur candidat de chaque sexe, $S = \{\arg \max_{\text{mâles}} b, \arg \max_{\text{femelles}} b\}$. Chaque itération résout la forme fermée sur le S courant et se ramifie :

- **Infaisable sur S** (le support est trop apparenté pour atteindre la borne k) : ajouter les candidats les moins apparentés, ceux de plus petit $(\mathbf{G}\mathbf{c})_j$, par paquet doublant $|S|$. Cette phase de faisabilité, lorsqu'elle se faisait un candidat à la fois, était le coût dominant ; le traitement par paquets a réduit le nombre total d'itérations d'environ un facteur trente, pour un optimum identique.
- **Une contribution est négative** : retirer tout i tel que $c_{S,i} \leq 0$, en les marquant tabous à la réintroduction jusqu'à la prochaine amélioration véritable, puis re-résoudre ; si un retrait vide un sexe, son meilleur candidat est réintroduit.
- **Sinon** : évaluer chaque candidat $j \notin S$ par son coût réduit $r_j = b_j - \mu_{\text{sexe}(j)} - 2\lambda(\mathbf{G}\mathbf{c})_j$, et ajouter le maximiseur si $r_j > \text{tol}$; si aucun n'est positif, arrêter. Un ajout véritable vide l'ensemble tabou.

Le problème étant convexe, un point réalisable où aucun candidat n’a de coût réduit positif satisfait les conditions de Karush–Kuhn–Tucker et constitue l’optimum global ; la correction n’est donc pas heuristique. L’ensemble tabou, conjugué à un seuil de dégénérescence calibré sur l’ordre de grandeur des coefficients, a empêché tout cyclage sur chaque instance résolue ici, sans que nous prouvions la terminaison finie en général. La forme fermée reste bien conditionnée en pratique : avec le ridge ε , $\kappa(\mathbf{G}_S)$ va d’environ 10 à 4000 aux supports opérationnels sur les panels réels (le plus élevé sur les lignées de blé autogames), et un bloc de génotypes exactement dupliqués — une $\mathbf{G}_S$ singulière sans le ridge — se résout tout de même en un optimum KKT réalisable, les doublons se partageant leur contribution à parts égales. L’évaluation par coût réduit vers une *unique* borne fixée k distingue la mise à jour du support de l’algorithme de la ligne critique, qui modifie l’ensemble actif de ± 1 le long d’un multiplicateur balayé pour tracer toute la frontière efficace.

Un plafond par candidat $\mathbf{c} \leq \mathbf{u}$ est absorbé par le même ensemble actif sans perte de la forme fermée. Une contribution libre qui dépasse son plafond est fixée à u_j et déplacée dans un ensemble *supérieur* U ; un candidat déjà dans U est relâché vers l’ensemble libre quand son coût réduit devient négatif. À U fixé, la résolution restreinte conserve sa structure — les contributions fixées entrent dans le second membre affine sous la forme $\mathbf{d} - \mathbf{A}_U \mathbf{u}_U$ et dans l’ellipsoïde actif comme un décalage sans matrice $\mathbf{G}_U \mathbf{u}_U$ plus une constante, laissant la même factorisation de Cholesky et la même équation quadratique scalaire. La résolution bornée se réduit à la résolution non bornée quand aucun plafond n’est actif, et atteint le même optimum qu’un solveur conique à point intérieur sur le programme conique $\mathbf{c} \leq \mathbf{u}$ correspondant, sous plafonds actifs.

2.5 Produits de parenté sans matrice

Tout produit de parenté complet est formé sans matérialiser $\mathbf{G}$:

$$\mathbf{G}\mathbf{c} = \varepsilon \mathbf{c} + \mathbf{Z}(\mathbf{Z}^\top \mathbf{c})/s,$$

deux produits matrice–vecteur contre la matrice de génotypes $n \times m$, en $O(n \cdot m)$ en temps et $O(n \cdot m)$ en mémoire pour $\mathbf{Z}$ contre $O(n^2)$ pour une $\mathbf{G}$ dense. La $\mathbf{G}_S$ restreinte n’est assemblée que sur le support. Les points d’entrée Rust packés dédiés conservent les appels entiers à deux bits (les trois valeurs entières n’en demandent pas plus), le centrage appliqué à la volée par des corrections rang-1, de sorte que son empreinte est $nm/4$ octets — encore $32\times$ sous un $\mathbf{Z}$ dense en $f64$; les résolutions packée et dense rendent le même optimum à la précision machine. Le lecteur Rust remappe les octets PLINK `.bed` sans développer une matrice entière. Les bindings R et Python livrés prennent actuellement $\mathbf{Z}$ dense et n’exposent pas cette voie packée. Sur le panel porcin à 52k SNP, le store occupe les 46,7 Mo du fichier contre 1,49 Go pour la matrice dense en $f64$, à support identique et gain concordant à 4×10^{-14} ; sur un panel 5000×50000 , mesurée par processus, la mémoire résidente de pointe est de 69 Mo contre 2,02 Go. Deux limites de la représentation méritent d’être énoncées. Les appels manquants sont gratuits : le quatrième code 2 bits en marque un et se décode en $2p_j$, l’imputation à la moyenne du marqueur que fait le lecteur dense, donc la concordance reste exacte. Les *dosages* imputés ne le sont pas : deux bits ne tiennent que des appels durs, et 0,127 % des valeurs de ce panel porcin sont fractionnaires, donc la voie packée les arrondit — ce qui déplace l’optimum de $1,8 \times 10^{-7}$ en gain sur un support identique de 23 individus, le prix de la représentation sur un vrai panel à dosages imputés. C’est l’élément habilitant à grand n , où la matrice dense $n \times n$ $\mathbf{G}$ peut être coûteuse à stocker ou à construire ($O(n^2 m)$) à construire ; ce n’est pas une accélération de boucle interne quand $m > n$, et l’avantage sur les références coniques est algorithmique — le petit ensemble actif — indépendant de ce choix.

2.6 Implémentation et reproductibilité

Le solveur est implémenté en Rust sur une pile purement Rust : algèbre linéaire dense (la factorisation de Cholesky et les remontées sur le support) via `faer`, sans dépendance système

BLAS/LAPACK, sans `unsafe`, et avec un générateur aléatoire reproductible amorcé pour les générateurs synthétiques. Un solveur conique à point intérieur (`clarabel`) n’est inclus que comme oracle indépendant de contre-vérification. La correction est attestée par des tests unitaires à certificat KKT sur une plage de bornes k — un résultat `Solved` est toujours réalisable, sur le budget et réparti par sexe — et par un accord inter-langage : sur une instance active, le `solve_sexed` en Rust reproduit un optimum de référence NumPy à une différence de gain de $1,5 \times 10^{-14}$ près. La compilation est conditionnée à `cargo fmt`, `cargo clippy -D warnings` et la suite de tests.

2.7 Données et références de comparaison

La méthode est évaluée sur trois panels de marqueurs publics : un panel de blé CIMMYT (BGLR, $n = 599$), un panel de porc PIC ($n = 3534$, 52k SNP, avec valeurs génétiques estimées réelles) et un panel de souris en population hétérogène (BGLR, $n = 1814$, avec sexe enregistré, 934 mâles et 880 femelles, l’indice de masse corporelle servant de critère de sélection). Pour chacun, $\mathbf{G}$ est la matrice de VanRaden avec régularisation ridge $\varepsilon = 10^{-5}$.

Une quatrième instance est dérivée du panel PIC et est la seule à porter simultanément une vraie valeur génétique et un sexe réel. Les données PIC sont livrées avec un pedigree, et un pedigree assigne le sexe sans ambiguïté : un animal figurant en colonne père est mâle, en colonne mère femelle. Le croisement avec les animaux génotypés restitue un sexe enregistré pour 1194 des 3534 (390 pères, 804 mères, aucun n’apparaissant comme les deux), tous porteurs d’une valeur génétique estimée réelle issue de l’évaluation associée (précision moyenne 0,811 sur le caractère 3). Les fréquences alléliques, et donc $\mathbf{G}$, sont recalculées sur les animaux retenus. Le sexe n’étant récupérable que pour les animaux devenus parents, ce sous-panel est un sous-ensemble post-sélection plutôt qu’un échantillon aléatoire de candidats — une propriété de la récupération, non un choix, qui porte sur l’interprétation du vecteur de contributions et non sur ce que mesure la comparaison. Les valeurs fournies avec ce panel provenant de l’évaluation propre au programme de sélection et n’étant pas des prédictions génomiques, le sous-panel est également traité avec un critère calculé comme le calcule la sélection génomique : un GBLUP ajusté par BGLR sur les phénotypes du caractère 3 du panel (3141 des 3534 animaux) contre sa propre matrice de parenté génomique, le critère étant lu comme la moyenne a posteriori de l’effet génétique. L’héritabilité génomique ajustée vaut 0,247, et 119 des 1194 animaux du sous-panel ne portent aucun phénotype propre : leur valeur provient donc uniquement des parentés génomiques. Ce contrôle de sensibilité du critère est ajusté sur le même échantillon ; aucun phénotype tenu à l’écart ni génération indépendante n’évalue sa précision prédictive. Les références de comparaison sont optiSel (R, le solveur à point intérieur Nesterov–Todd `cccp` ; la référence exacte du domaine), Clarabel (la contre-vérification conique) et AlphaMate (évolution différentielle en Fortran ; exécuté depuis son binaire Linux publié, sous émulation — aucune compilation macOS n’est distribuée, et la source ne peut être construite sans Intel MKL, que son `src/CMakeLists.txt` lie inconditionnellement et qui n’a pas de version Apple Silicon). optiSel est épinglé à la version 2.1.0, sa dernière : le paquet a été archivé du CRAN le 17 septembre 2026 « faute de corrections apportées à temps », toutes les saveurs de vérification — `r-release-macos-arm64` comprise — étant rapportées OK le jour même, de sorte que rien ici n’est affecté ; la 2.1.0 reste installable depuis l’archive du CRAN, d’où les scripts de reproduction la récupèrent désormais. Les générateurs de code pour programmes convexes paramétrés — CVXGEN (Mattingley & Boyd 2012), CVXPYgen (Schaller et al. 2022) et, en Rust au-dessus du même Clarabel, `cvxgenrust` (Zhu & Boedeker 2026) — émettent ce même programme conique générique et héritent de sa factorisation en $O(n^3)$: ils sont complémentaires d’un solveur exploitant la structure, non des alternatives ; nous utilisons le dernier comme vérification indépendante de la référence conique à la section 3.2. Réserves reportées à la Discussion : le sexe enregistré est réel pour le panel souris et pour le sous-panel porcin sexué, tandis que le blé (autogame) et le panel porcin complet reçoivent une répartition équilibrée arbitraire ; le critère de sélection $\mathbf{b}$ est une valeur génétique estimée réelle sur les panels porcins et un phénotype enregistré en tenant lieu sur

le blé et la souris ; et le Tableau 2 rapporte deux chronométrages de support-first face à optiSel — l'ensemble actif avec **G** dense fournie (le régime d'optiSel, un prototype NumPy) et le solveur sans matrice en Rust livré, de bout en bout — séparant ainsi l'avantage algorithmique de ce que l'outil livré paie réellement, au lieu de les confondre.

3 Résultats

3.1 Exactitude

Pour la formulation convexe, un certificat KKT réalisable avec stationnarité, multiplicateurs non négatifs et complémentarité prouve l'optimalité globale ; l'implémentation les vérifie à tolérance près. Cela ne prouve pas la terminaison finie en toute instance. Le solveur Rust sexué concorde avec une référence NumPy à $1,5 \times 10^{-14}$ en gain sur une instance active ; les contrôles Clarabel synthétiques ont des écarts sous 10^{-8} aux bornes testées. Les plafonds individuels actifs sont également vérifiés face au programme conique correspondant.

L'essai historique optiSel **Cattle** ($n = 268$, sexe fourni, phénotypes simulés) sélectionnait les mêmes 36 candidats à 3×10^{-4} près en contributions ; la parenté réalisée différait (0,0576 contre la borne 0,0578). Sur la souris aux cibles de proxy 2/1/0,5%, optiSel utilisait environ 98,2/95,3/85,7% de la borne, avec des écarts de gain 0,000306/0,000865/0,002608. Sur le blé, ils étaient plus grands : 0,013498/0,046743/0,248247. Ce sont les sorties des configurations archivées à la borne demandée, pas des optima identiques ; l'accord se mesure séparément à coancestrie réalisée appariée. Le script révisé `exactness_operational.R` réalise ce contrôle distinct ; la nouvelle voie de temps partage la borne exportée et l'empreinte des entrées entre implémentations.

COMA (Endelman 2025), via CVXR/ECOS, fournit une contre-vérification historique du domaine sur deux panels. Les écarts de gain rapportés sont $1,3 \times 10^{-8}$ – $2,9 \times 10^{-5}$ et ceux de contributions $7,5 \times 10^{-7}$ – $2,3 \times 10^{-5}$; les effectifs de support dépendent du seuil de rapport 10^{-6} . L'utilisation de la borne rapportée vaut 100,00001–100,00968%, contre 100,00000% pour support-first. Les temps 2,6–51 s contre 0,07–8,7 s incluent langues et canonicalisations différentes. Ces résidus et temps concernent les configurations testées, sans classer la précision des classes de solveurs. La forme fermée utilise toujours une arithmétique de précision finie et des tolérances de certificat.

3.2 Vitesse face à un solveur conique généraliste

Cette comparaison tranche en outre une question que la confrontation à optiSel ne peut pas trancher : les deux solveurs sont ici en Rust, appelés dans un même processus sur les mêmes données générées, de sorte qu'aucune différence de langage ne peut être prise pour une différence algorithmique. Sur des instances génomiques synthétiques à borne relative plus serrée, réglée pour un support ≈ 100 , l'avantage de support-first sur le solveur conique Clarabel croît fortement avec n (Tableau 1) : d'environ $1,6\times$ à $n = 1000$ à $\sim 716\times$ à $n = 10000$, où la voie conique prend 38 minutes et support-first 3,2 s. Le passage à l'échelle est structurel — Clarabel factorise un système KKT dense en $O(n^3)$ à chaque itération de point intérieur (2,1 s à 2290 s sur le balayage), tandis que support-first effectue ~ 150 à 170 produits matrice–vecteur, chacun en $O(n \cdot m)$, son temps ne croissant que doucement (1,4 s à 3,2 s). À borne lâche le support s'effondre à une poignée et le facteur atteint $\sim 76000\times$; ce résultat historique à cap lâche est distinct de la table à cap plus serré. La référence n'est pas bridée : un solveur généré indépendamment par `cvxgenrust` à partir d'un modèle CVXPY de six lignes du même programme conique reproduit le même optimum (gain à 2×10^{-8} près) à $n = 500$ et $n = 1000$, au coût par itération d'un binaire de ce crate (103 ms par itération pour les deux, machine au repos) et 13 % au-dessus du binaire avec lequel la table 1 a été mesurée — pourvu que le facteur soit déclaré avec son motif de parcimonie triangulaire, auquel cas le programme généré ne porte que le triangle et n'exige aucun réglage du solveur ; déclaré

comme paramètre dense, il porte $n(n-1)/2$ zéros explicites et est 3 à $5\times$ plus lent à moins de régler Clarabel pour les éliminer. Deux bornes sur cette ligne : le coût par itération d’une seule et même compilation de Clarabel diffère de $\sim 14\%$ entre binaires de ce crate, ce qui est la précision du rapport à $n = 1000$ et sans conséquence dès $n = 5000$; et la génération de code elle-même demande 8 Go de mémoire et 43 Mo de source générée à $n = 1000$, croissant en n^2 — le régime embarqué, non celui des populations.

Tab. 1 : support-first contre Clarabel à borne relative plus serrée, avec accord en gain à $\sim 10^{-8}$ près, un balayage en série sur machine au repos. Les deux solveurs sont en Rust dans un même processus sur la même instance synthétique ($m = 20000$, borne $k = 0.017 \times \text{mean}(\text{diag } \mathbf{G}_0)$ réglée pour un support ≈ 100 ; le N_e de contributions atteint n’est pas archivé) : la comparaison ne comporte aucun biais de langage.

n	m	Clarabel ¹	support-first	accélération
1000	20000	2,22 s	1,43 s	1,6×
2000	20000	16,2 s	1,90 s	9×
5000	20000	308 s	2,36 s	131×
10000	20000	2298 s	3,21 s	716×

¹ toute la voie conique : construction de $\mathbf{G}$, sa factorisation de Cholesky, l’assemblage du cône et la résolution à point intérieur. La résolution à point intérieur seule fait 2,1 / 15,9 / 306 / 2290 s. La colonne support-first est son exécution entière, puisqu’il ne construit rien.

Deux réserves. Les cellules à grand n sont des exécutions uniques (Clarabel demande 38 minutes à $n = 10000$) ; le côté support-first est déterministe. Et l’avantage est entièrement piloté par n : à $n = 1000$ les temps sont du même ordre, car à $m = 20000$ les ~ 150 produits de support-first en $O(nm)$ égalent la résolution $O(n^3)$ de Clarabel à petit n — l’écart ne s’ouvre que lorsque n^3 dépasse $n \cdot m$.

3.3 Vitesse face à l’outil du domaine optiSel

Une comparaison au domaine complète le benchmark conique. Le Tableau 2 distingue le prototype NumPy recevant $\mathbf{G}$ dense, avec produits indépendants de m , de la voie Rust sans matrice livrée, coûtant $O(nm)$ par produit. Aux cibles historiques approximativement appariées $\Delta F_{\text{proxy}} = 1\%$ ($N_e \approx 50$), le prototype est environ 12 à 133× plus rapide qu’optiSel ; Rust sans matrice est environ 3× plus rapide sur le blé, 6× sur la souris, mais 0,3× aussi rapide sur le porc. Les calibrations historiques étaient indépendantes : les colonnes ne garantissent pas une borne k ou un optimum identique. L’accord numérique des solveurs s’évalue séparément à coancestrie réalisée appariée et par les tests de certificat.

Tab. 2 : Temps historiques à cibles approximativement appariées du proxy $\Delta F_{\text{proxy}} = 1\%$ ($N_e \approx 50$), Apple M4 Max avec R/OpenBLAS 0.3.33. *Algorithme* : prototype NumPy, $\mathbf{G}$ dense précalculée, résolution seule. *Sans matrice* : Rust livré, génotypes denses, résolution de bout en bout sans $\mathbf{G}$ dense. Les colonnes calibraient les bornes indépendamment : ni bornes ni optima identiques ne sont garantis. $|S|$ est le support support-first rapporté. Temps historiques enregistrés : médianes de trois runs pour le prototype, mais runs individuels dans les scripts R/optiSel et sans matrice ; les répétitions brutes ne sont pas archivées. Ratios calculés sur les temps archivés non arrondis ; les temps affichés sont arrondis.

jeu de données	n	m	$ S $	algorithme ¹	sans matrice ²	optiSel	algo. ³	s.mat. ³
blé CIMMYT (GRM réelle)	599	1279	109	0,055 s	0,214 s	0,667 s	12×	3×
souris HS (GRM réelle, sexe réel)	1814	10346	89	0,061 s	1,46 s	8,06 s	133×	6×
porc PIC, sexué (VGE & sexe réels) ⁴	1194	52843	110	0,078 s	10,06 s	2,77 s	35×	0,3×

¹ ensemble actif avec $\mathbf{G}$ dense précalculée, solve seul (prototype NumPy, pas une résolution dense Rust livrée) ; ² sans matrice, ne forme aucun $\mathbf{G}$, bout en bout depuis les génotypes (Rust livré) ;

³ accélération sur optiSel, pour chacune des deux colonnes ; ⁴ le sous-panel PIC portant une VGE réelle *et* un sexe réel (Méthodes).

Le point de fonctionnement fixe l'échelle. À $\Delta F_{\text{proxy}} = 2\%$ ($N_e = 25$) les supports sont plus petits (47 à 57) et les deux voies plus rapides — sans matrice $9\times$ (blé) à $34\times$ (souris), l'algorithme 40 à $482\times$; à $\Delta F_{\text{proxy}} = 0,5\%$ ($N_e = 100$) les supports montent à 182–226 et la voie sans matrice atteint la parité voire pire (souris $0,9\times$, porc $18\times$ plus lent), tandis que l'algorithme reste 2 à $20\times$ devant. La borne lâche qu'une version antérieure de ce travail rapportait place le point de fonctionnement à $N_e \approx 10$ ($\Delta F_{\text{proxy}} \approx 5\%$), une concentration de contributions supérieure aux cibles opérationnelles ici ; les accélérations y sont plus grandes mais elles appartiennent à un autre point de fonctionnement. Aux trois points, l'algorithme — recevant $\mathbf{G}$, évaluation en $O(n^2)$ — est plus rapide que la colonne sans matrice sur chaque panel, puisque avec $m > n$ sur chacun, $O(n^2) < O(nm)$; la colonne sans matrice est le prix pour garder les génotypes denses en évitant la parenté dense. Ces ratios historiques portent sur des cibles de proxy approximativement appariées ; l'accord numérique se teste séparément à borne commune ou coancestrie réalisée appariée.

Le sous-panel porcin sexué est la seule instance portant à la fois une vraie valeur génétique et un sexe réel, donc la plus proche ici de ce qu'exécute un programme de sélection. Le sexe n'y est pas supposé mais lu dans le pedigree fourni avec les données PIC — père $\Rightarrow$ mâle, mère $\Rightarrow$ femelle —, ce qui résout 1194 des 3534 animaux génotypés (390 pères, 804 mères, aucun n'apparaissant comme les deux), tous porteurs d'une VGE réelle de l'évaluation associée à précision moyenne 0,811. C'est aussi la ligne la moins favorable à la voie sans matrice ($m/n \approx 44$), ce qui la rend précisément informative : elle montre la pénalité sans matrice à son pire *et* l'avantage algorithmique qui y survit — $35\times$ sur optiSel avec $\mathbf{G}$ fournie, contre le $0,3\times$ sans matrice. Le run historique support-first sélectionnait 49 mâles et 61 femelles à $\Delta F_{\text{proxy}} = 1\%$ (support 110). Répéter la comparaison avec une vraie GEBV — un GBLUP ajusté sur les phénotypes et la matrice de parenté propres au panel, corrélé à 0,64 avec la VGE fournie — a donné un accord numérique similaire sur cette instance, sans valider la prédiction ni la robustesse à tout critère. Réserves reportées à la Discussion : sur le blé et la souris le critère est un phénotype enregistré plutôt qu'une valeur génétique, le GBLUP porcin supplémentaire est un critère génomique ajusté sur le même échantillon ; et le sous-panel porcin sexué, constitué des animaux devenus parents, est un sous-ensemble post-sélection.

Le critère GBLUP supplémentaire est ajusté sur les phénotypes et la parenté du même panel. La comparaison a donné un accord numérique similaire sur cette instance. Ce seul contrôle de sensibilité n'est pas une validation prédictive ; il ne garantit ni précision hors échantillon ni robustesse à toutes les valeurs génétiques utilisées comme critère.

3.4 Comparaison avec l'heuristique AlphaMate

AlphaMate optimise un problème apparenté mais distinct — l'allocation discrète des accouplements par un algorithme évolutionnaire stochastique — de sorte que ce n'est pas une confrontation à armes égales. Nous évaluons *son* vecteur de contributions dans notre métrique et comparons, à coancestrie de groupe appariée, uniquement sur la relaxation à contributions continues commune aux deux méthodes. Sur cette relaxation, l'optimum exact de support-first a un gain supérieur à celui du vecteur d'AlphaMate en chaque point de sa frontière (Tableau 3) : une faible marge au compromis à 45° ($\Delta\text{gain} +0,004$) et des marges plus grandes aux coins — comme il se doit d'un optimum convexe exact face à une heuristique stochastique portant en outre des contraintes d'accouplement entières. C'est un contrôle de cohérence — support-first est exact sur la relaxation commune — et non une victoire sur AlphaMate à sa propre tâche, l'allocation discrète des accouplements, que support-first n'effectue pas. AlphaMate s'est en outre révélé nettement fragile sur données génomiques réelles : une exécution réussie a exigé six configurations et trois contournements — plafonner les accouplements sous n ; restaurer l'ensemble complet des parents, pour éviter une faute de segmentation à l'initialisation déclenchée par le nombre réduit de parents ; et décaler positivement le critère, pour défaire une inversion de signe sur les VGE centrées et négatives —

alors que support-first et optiSel ont tourné sans modification. Nous ne rapportons pas de temps d'exécution d'AlphaMate : le seul binaire ayant tourné ici était une compilation x86 sous émulation, dont le temps n'est pas comparable.

Tab. 3 : Gain génétique à coancestrie de groupe appariée $\mathbf{c}^\top \mathbf{K} \mathbf{c}$, où $\mathbf{K} = \mathbf{G}/2$ est la matrice de coancestrie, panel souris ; le vecteur de contributions de chaque méthode évalué dans la même métrique.

coancestrie de groupe $\mathbf{c}^\top \mathbf{K} \mathbf{c}$	AlphaMate	support-first	Δ gain
0,000272 (coancestrie min. AlphaMate)	-0,45885	-0,37843	+0,080
0,001317 (optimum 45° AlphaMate)	-0,35748	-0,35318	+0,004
0,007574 (critère max. AlphaMate)	-0,34066	-0,32240	+0,018

3.5 Passage à l'échelle et avantage sans matrice

Le stockage ne dépend pas du point de fonctionnement, contrairement aux temps de résolution. Le balayage structuré à $m = 1000$ et proxy $\Delta F_{\text{proxy}} = 1\%$ donne un support d'environ 80–103 ; le run publié $n = 40000$ a pris 0,646 s. La construction dense historique de 3,4 s à $n = 30000$ est une mesure séparée ; l'ancien facteur 113× utilisait une résolution à cap lâche d'un autre point et n'est pas une accélération opérationnelle. La construction dense était omise au-delà de $n = 30000$ par politique. À $n = 40000$, $8n^2$ octets représentent 11,9 Gio, contre 0,30 Gio pour $\mathbf{Z}$ dense (Figure 1) ; ces empreintes n'attestent pas une erreur mémoire observée. D'autres formulations sans matrice peuvent également éviter $\mathbf{G}$ dense. Une borne absolue peut se relâcher avec n , d'où le point du proxy fixé dans cette expérience.

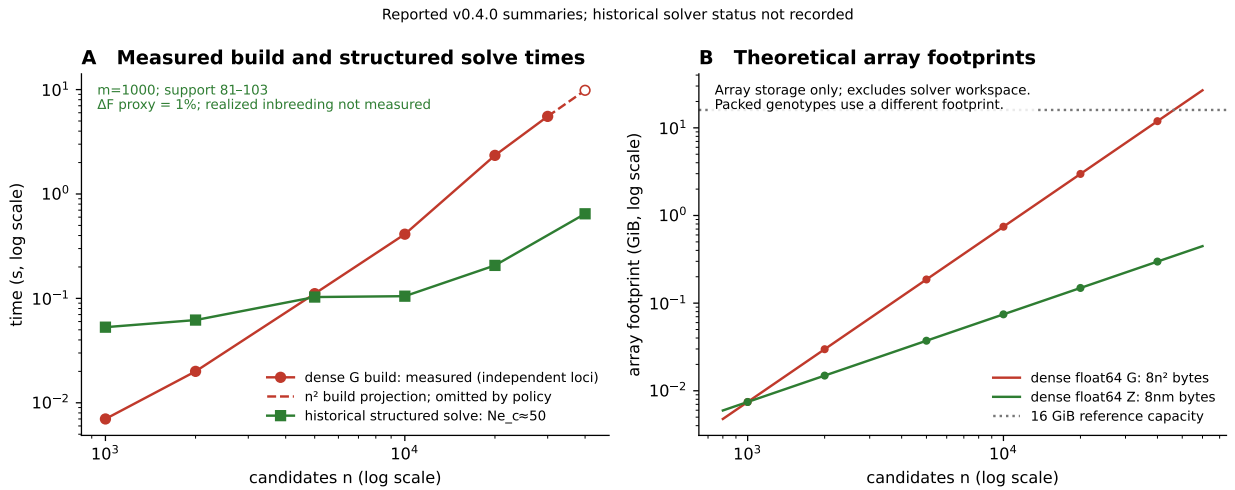


Fig. 1 : Passage à l'échelle structuré au proxy $\Delta F_{\text{proxy}} = 1\%$ ($N_e = 50$), $m = 1000$. **(A)** Support d'environ 80–103 ; run sans matrice publié à $n = 40000$ en 0,646 s. Construction dense omise au-delà de $n = 30000$ par politique du benchmark. **(B)** Stockages calculés : 11,9 Gio pour $\mathbf{G}$ dense contre 0,30 Gio pour $\mathbf{Z}$ dense à $n = 40000$. Ce ne sont pas des échecs mémoire mesurés.

3.6 Comportement du support

Le support participe empiriquement à l'avantage de vitesse. Les cibles du proxy $\Delta F_{\text{proxy}} = 2/1/0,5\%$ correspondent aux ratios de participation 25/50/100, sans équivalence avec une consanguinité génomique mesurée ni une taille efficace démographique. La souris a des supports 47/89/182 à ces points et environ 1163 près de la coancestrie minimale. Dans le balayage structuré historique au proxy 1% fixé, le support vaut environ 80–103 quand n passe de 1000 à 40000. Fixer

le ratio de participation ne borne pas supérieurement le support brut : $N_e \leq |S|$ fournit seulement une borne inférieure. Ces observations portent sur les populations testées. Une borne absolue peut devenir moins restrictive lorsque la plage de coancestrie réalisable change avec n ; un support plat à cette borne ne prouve pas une indépendance en taille de population. La Figure 2 rapporte support et temps des trois panels selon leurs cibles de proxy.

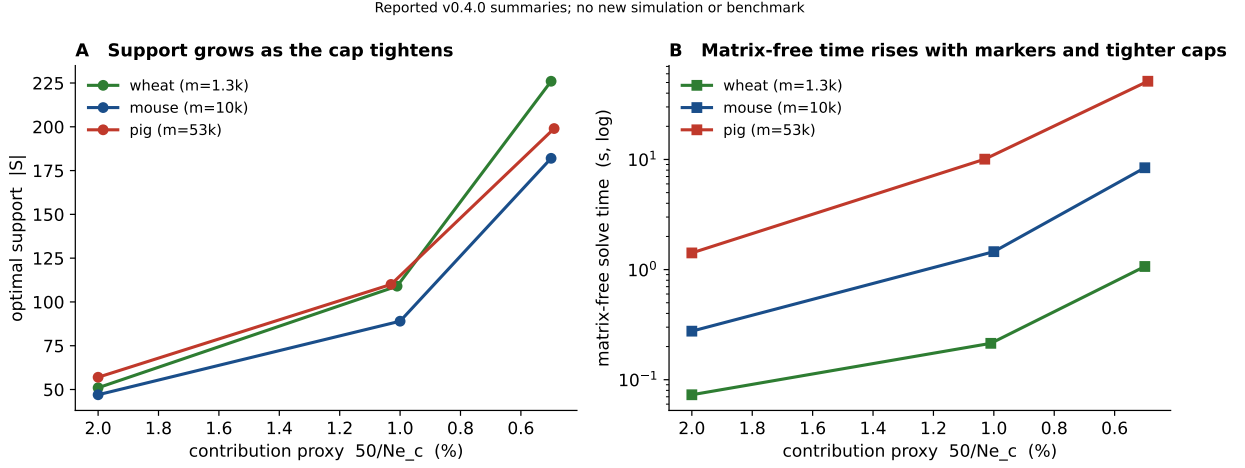


Fig. 2 : La frontière du point de fonctionnement sur les trois panels réels. **(A)** Le support observé croît quand la borne de coancestrie se resserre (plus petit ΔF_{proxy} , plus grand $N_e \approx 1/(2\Delta F_{\text{proxy}})$). **(B)** Le temps observé de résolution sans matrice croît quand la borne se resserre et, à ΔF_{proxy} fixé, avec le nombre de marqueurs m — porc (53k SNP) au-dessus de souris (10k) au-dessus de blé (1,3k) — le coût $O(nm)$ par itération rendu visible.

3.7 Validation multi-générations

Les résultats mono-génération établissent l’exactitude et le coût ; la question de sélection est de savoir si support-first appliqué en récurrent se comporte comme doit le faire la sélection à contribution optimale au fil des générations. Nous avons simulé 12 générations de sélection récurrente sur des populations AlphaSimR (120 fondateurs coalescents, un caractère additif, 3 répétitions), chaque génération appelant le binding R livré pour résoudre l’OCS à $\Delta F_{\text{proxy}} = 1\%$ ($N_e = 50$) sur les génotypes courants et croiser proportionnellement aux contributions rendues, face à la sélection par troncature (les 25 meilleurs de chaque sexe par valeur génétique vraie) comme référence indifférente à la diversité. Support-first conserve nettement plus de diversité — l’hétérozygotie observée tombe à 0,246 à la génération 12 contre 0,221 pour la troncature (toutes deux depuis 0,290), une perte $\sim 40\%$ plus lente — tout en *soutenant* le gain : il égale la troncature tôt et la dépasse légèrement tard (11,9 contre 11,6 à la génération 12, Figure 3), ce qui concorde avec une réponse durable par préservation de diversité. La cible ΔF_{proxy} est le proxy de contributions ; l’hétérozygotie est la diversité génomique mesurée séparément. Ces résultats sont ceux de la simulation historique archivée.

Une nouvelle vérification du 11 octobre 2026, avec trois répétitions et sauvegarde des fondateurs et de l’état aléatoire, donne à la génération 12 un gain moyen de 11,46 pour OCS contre 11,51 pour la troncature, avec une hétérozygotie observée de 0,234 contre 0,213. Elle reproduit la préservation de diversité, mais pas le léger avantage historique de gain final. Cet avantage dépend donc de la simulation rapportée et ne constitue pas un résultat universel. Les résumés et sauvegardes des fondateurs sont fournis dans `research/repro/verified/2026-10-11/` ; la figure historique est conservée séparément. Ces sauvegardes répondent à une limitation observée de la graine MaCS dans AlphaSimR 2.1.0 : la graine seule ne régénérerait pas des fondateurs identiques.

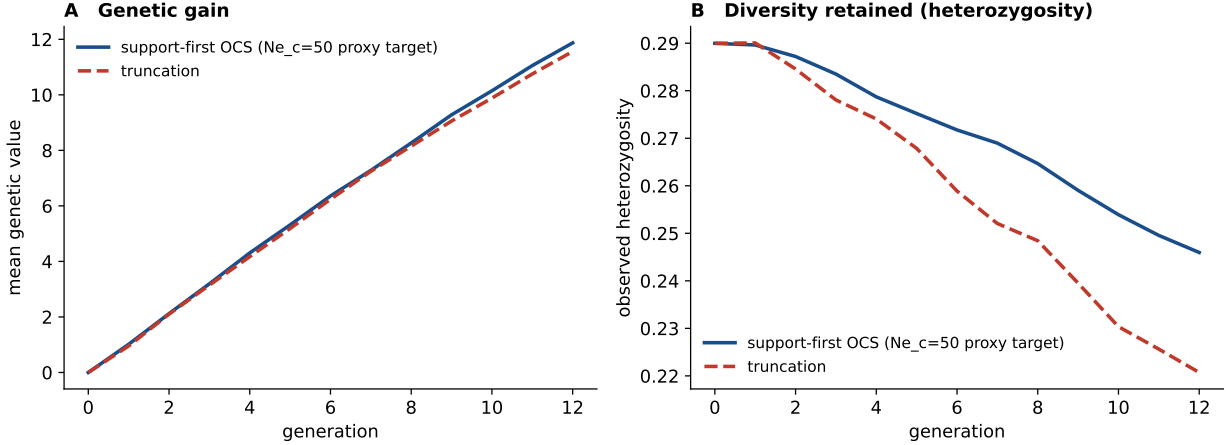


Fig. 3 : Simulation historique archivée : douze générations de sélection récurrente sur populations Alpha-SimR (3 répétitions) : support-first OCS à $\Delta F_{\text{proxy}} = 1\%$ contre troncature. **(A)** Gain génétique — OCS égale la troncature et la dépasse tard. **(B)** Hétérozygotie observée — OCS la conserve nettement mieux (0,246 contre 0,221 à la génération 12), avec la trajectoire de gain présentée en (A).

4 Discussion

Les résultats distinguent le prototype NumPy recevant $\mathbf{G}$ dense de la voie Rust sans matrice livrée. Aux cibles approximativement appariées du proxy $\Delta F_{\text{proxy}} = 1\%$, le prototype est environ $12 \times$ plus rapide qu’optiSel ; Rust est environ $3 \times$ plus rapide sur le blé, $6 \times$ sur la souris et $0,3 \times$ aussi rapide sur le porc sexué. La voie sans matrice paie $O(nm)$ par produit pour éviter la mémoire de parenté dense. L’instance structurée $n = 40000$, $m = 1000$ démontre 0,646s et une réduction calculée de stockage de $40 \times$, pas l’échec d’un solveur dense. À entrées et configuration numérique fixées, l’algorithme est reproductible à tolérance près ; le dernier chiffre n’est pas garanti entre machines, bibliothèques et configurations.

Les ingrédients de la méthode sont individuellement classiques, et nous n’en revendiquons que la synthèse. Le suivi par ensemble actif d’un petit ensemble de travail descend de l’algorithme de la ligne critique pour la sélection moyenne-variance long-only ; la forme fermée par support est un résultat de valeur propre contrainte ; le produit sans matrice est standard en prédiction génomique. La contribution est leur combinaison, spécialisée à l’OCS : une génération de colonnes par coût réduit qui fait croître un support restreint vers une unique borne de coancestrie, avec le produit de parenté évalué sans matrice de sorte que le coût suit le support et le nombre de marqueurs plutôt que la matrice $n \times n$. Située face aux travaux antérieurs les plus proches, ce n’est ni la parcimonie de la matrice de pedigree d’un solveur OCS à point intérieur, ni les résolutions denses sur toute la population des outils de référence. Deux comparaisons plus proches que nous n’avons pas exécutées, et que nous signalons plutôt que d’écarter. Gencont2 (Dagnachew & Meuwissen 2016) est un Newton–Raphson sur les conditions lagrangiennes à $O(n^2)$ par itération — le même coût par itération que notre pricing —, donc à coût égal c’est le test juste de savoir si un ensemble actif croissant converge en moins d’itérations qu’un Newton sur l’ensemble complet ; sa distribution est liée aux relations généalogiques et nous ne l’avons pas réimplémenté sur une $\mathbf{G}$ génomique, ce qui est le benchmark le plus informatif encore ouvert. Le solveur ADMM/JuMP de Waldmann (2025) forme $\mathbf{G}$ explicitement et porte le coût de stockage dense dans cette formulation, mais est parfaitement benchmarkable sur les petits panels ici ; nous laissons les deux à des travaux futurs plutôt que d’invoquer une incomparabilité.

Plusieurs limites bornent ces résultats. Le critère est une EBV fournie sur le porc et un phénotype sur le blé et la souris. Le critère GBLUP supplémentaire du porc sexué est ajusté sur le même échantillon : c’est un contrôle de sensibilité, pas une validation prédictive, et il ne clôt pas la ques-

tion du critère. Le sexe est enregistré pour la souris et récupérable pour les porcs devenus parents ; ceux-ci constituent un sous-ensemble post-sélection, tandis que blé et porc complet reçoivent des répartitions arbitraires. Les contributions sont illustratives, pas des recommandations de sélection.

Les performances portent sur des implémentations et points particuliers. La colonne dense est un prototype NumPy dont les temps ne se transfèrent pas au Rust livré. La voie Rust sans matrice est plus lente qu’optiSel sur le porc à $0,3\times$ la vitesse. Le packing Rust dédié réduit le stockage de génotypes d’environ $32\times$ mais était plus lent au chronométrage porcin historique (4,6s contre 2,85s en *f64* dense) ; R et Python prennent encore des entrées denses. Le packing gère appels entiers et imputation moyenne des manquants, avec la sensibilité rapportée à l’arrondi des dosages fractionnaires.

La petite taille du support opérationnel est empirique. Sans ridge ni plafonds supérieurs effectifs, fixer $\mathbf{Z}^\top \mathbf{c}$ et $\mathbf{b}^\top \mathbf{c}$ d’un optimum atteint définit une tranche optimale bornée avec un sommet de support au plus $\text{rang}([\mathbf{A}; \mathbf{Z}^\top; \mathbf{b}^\top]) \leq q + r + 1$. Aucune dualité n’est requise et rien ne garantit que support-first renvoie ce sommet. Avec plafonds finis, seules les coordonnées libres sont bornées ; celles au plafond positif comptent aussi. $u_i = 2/n$ et $\mathbf{G} = 0$ imposent au moins $\lceil n/2 \rceil$ contributeurs. La borne de rang peut être vide sur les panels à nombreux marqueurs ; avec $\varepsilon > 0$, $\mathbf{G} = \varepsilon \mathbf{I}$ impose un support plein si $\varepsilon/n < k < \varepsilon/(n-1)$. Une borne conditionnelle sous structure réaliste reste ouverte. Le proxy de contributions indexe la croissance observée, sans mesurer la consanguinité génomique réalisée.

Enfin, le solveur gère une contrainte quadratique de parenté, des plafonds individuels et budgets sexuels, pour des contributions continues. Plusieurs contraintes quadratiques et l’allocation entière des accouplements restent des extensions.

Tous les panels de marqueurs testés sont publics ; les instances structurées de $n = 10000$ à 40000 sont synthétiques. Elles établissent le comportement de calcul et les empreintes de ces générateurs, pas l’utilisation dans une évaluation nationale. Une population de production appropriée, des critères réalistes et une évaluation prédictive indépendante étendraient cette validation.

Chaque limite indique une extension. Le cœur en forme fermée par support admet déjà plus d’une contrainte d’égalité via la même élimination $\mathbf{P} = \mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top$. Avec plusieurs plafonds quadratiques actifs, la stationnarité fait intervenir $\mathbf{H} = \sum_j \lambda_j \mathbf{G}_j$; son inverse, lorsqu’il existe, dépend de multiplicateurs couplés. Cela requiert en général une méthode numérique distincte ; la forme fermée scalaire ne s’étend pas directement. Si la formulation discrète d’allocation des accouplements est contenue dans l’ensemble réalisable continu, le maximum continu donne une borne supérieure valide du gain ; son caractère serré n’est pas établi. Une couche d’arrondi ou de séparation et évaluation (branch-and-bound) pourrait utiliser cette borne pour traiter l’allocation discrète. Le basculement sans matrice en m versus n mérite une caractérisation explicite, tout comme la borne de support elle-même. Et la méthode devrait être validée sur des populations un ordre de grandeur plus grandes, où l’écart mémoire passe de visible à décisif ; le seul contrôle GBLUP sur le même échantillon laisse ouvertes les validations indépendantes du critère et des prédictions.

Support-first fait une chose — la sélection à contribution optimale exacte, à contrainte unique, continue — à une réduction de stockage et des performances mesurées dans un workflow reproductible. Cette affirmation étroite et vérifiée, et les extensions ouvertes qu’elle invite, constituent la contribution.

Conflits d’intérêts

L’auteur ne déclare aucun conflit d’intérêts.

Financement

Ce travail n’a bénéficié d’aucun financement externe.

Disponibilité des données et du code

Le solveur et les scripts sont disponibles à <https://github.com/Adelagric/ocs-rs> (licence MIT), avec une version historique archivée à <https://doi.org/10.5281/zenodo.20746987>. La voie locale révisée `research/repro/repro.sh` comprend contrôles du solveur, comparaisons opérationnelles R/support-first, passage à l’échelle structuré, simulation multi-génération et figures. Une exécution par défaut signale explicitement les dépendances ou données manquantes et peut être partielle ; une étape disponible qui échoue donne un code non nul. `REPRO_STRICT=1` exige toutes les étapes automatiques ; `REPRO_FULL=1` active aussi le grand balayage Clarabel ($n = 10000$: environ 38 minutes dans le run historique à cap plus serré). AlphaMate reste un workflow manuel distinct avec binaire émulé ; sa table n’est pas promise par une seule commande automatique. Ces corrections locales ne constituent pas encore une nouvelle version archivée. Panels : BGLR (blé/souris) et Cleveland, Hickey & Forni (2012) (PIC porc, DOI 10.1534/g3.111.001453).

Références

Méthodes et outils OCS

1. **Meuwissen, T.H.E. (1997).** Maximizing the response of selection with a predefined rate of inbreeding. *Journal of Animal Science* 75(4) :934–940. DOI 10.2527/1997.754934x.
2. **Dagnachew, B.S. & Meuwissen, T.H.E. (2016).** A fast Newton–Raphson based iterative algorithm for large scale optimal contribution selection. *Genetics Selection Evolution* 48(1) :70. DOI 10.1186/s12711-016-0249-2.
3. **Pong-Wong, R. & Woolliams, J.A. (2007).** Optimisation of contribution of candidate parents to maximise genetic gain and restricting inbreeding using semidefinite programming. *Genetics Selection Evolution* 39(1) :3–25. DOI 10.1186/1297-9686-39-1-3.
4. **Wellmann, R. (2019).** Optimum contribution selection for animal breeding and conservation : the R package optiSel. *BMC Bioinformatics* 20(1) :25. DOI 10.1186/s12859-018-2450-5.
5. **Gorjanc, G. & Hickey, J.M. (2018).** AlphaMate : a program for optimizing selection, maintenance of diversity and mate allocation in breeding programs. *Bioinformatics* 34(19) :3408–3411. DOI 10.1093/bioinformatics/bty375.
6. **Endelman, J.B. (2025).** Optimal contribution selection for any number of parents. *Genetics* 229(2) :iyae193. DOI 10.1093/genetics/iyae193. Logiciel : <https://github.com/jendelman/COMA> (v0.20, GPL-3) — le second solveur exact du domaine croisé à la section 3.1.
7. **Waldmann, P. (2025).** Genomic optimum contribution selection and mate allocation using JuMP. *Bioinformatics Advances* 5(1) :vbaf259. DOI 10.1093/bioadv/vbaf259.
8. **Yamashita, M., Mullin, T.J. & Safarina, S. (2018).** An efficient second-order cone programming approach for optimal selection in tree breeding. *Optimization Letters* 12(7) :1683–1697. DOI 10.1007/s11590-018-1229-y ; arXiv :1506.04487. — Exploite la parcimonie de l’inverse du pedigree, pas le support de la solution.

Parenté génomique et produits sans matrice

8. **VanRaden, P.M. (2008).** Efficient methods to compute genomic predictions. *Journal of Dairy Science* 91(11) :4414–4423. DOI 10.3168/jds.2007-0980.
9. **Legarra, A. & Misztal, I. (2008).** Technical note : Computing strategies in genome-wide selection. *Journal of Dairy Science* 91(1) :360–366. DOI 10.3168/jds.2007-0403.

Filiation ensemble actif et forme fermée (antériorité reconnue)

10. **Markowitz, H. (1956).** The optimization of a quadratic function subject to linear constraints. *Naval Research Logistics Quarterly* 3(1–2) :111–133. DOI 10.1002/nav.3800030110. — L’algorithme de la ligne critique.
11. **Gander, W., Golub, G.H. & von Matt, U. (1989).** A constrained eigenvalue problem. *Linear Algebra and its Applications* 114–115 :815–839. DOI 10.1016/0024-3795(89)90494-1. — La forme fermée par support (équation séculaire).

Logiciels et données

12. **Goulart, P.J. & Chen, Y. (2024).** Clarabel : an interior-point solver for conic programs with quadratic objectives. arXiv :2405.12762. (Version évaluée par les pairs : *Mathematical Programming Computation*, 2026, DOI 10.1007/s12532-026-00320-7.) — Utilisé uniquement comme oracle de contre-vérification indépendant.
13. **Mattingley, J. & Boyd, S. (2012).** CVXGEN : a code generator for embedded convex optimization. *Optimization and Engineering* 13(1) :1–27. DOI 10.1007/s11081-011-9176-9.
14. **Schaller, M., Banjac, G., Diamond, S., Agrawal, A., Stellato, B. & Boyd, S. (2022).** Embedded code generation with CVXPY. *IEEE Control Systems Letters* 6 :2653–2658. DOI 10.1109/LCSYS.2022.3173209.
15. **Zhu, H. & Boedeker, J. (2026).** Generation of custom solvers in Rust for convex optimization. arXiv :2609.13875. Logiciel : <https://github.com/dxogrp/cvxgenrust> (v0.1.0). — `cvxgenrust`, utilisé pour reconstruire indépendamment la référence Clarabel à partir de la formulation CVXPY (section 3.2).
16. **Quiñones, S.** `faer` : a linear-algebra library for the Rust programming language. Logiciel, version 0.24.0 (épinglée dans `Cargo.toml`) ; <https://github.com/sarah-quinones/faer-rs>. (Soumission JOSS en cours d’examen ; pas de DOI publié à ce jour.)
17. **Pérez, P. & de los Campos, G. (2014).** Genome-wide regression and prediction with the BGLR statistical package. *Genetics* 198(2) :483–495. DOI 10.1534/genetics.114.164442. — Source des panels blé (CIMMYT) et souris.
18. **Cleveland, M.A., Hickey, J.M. & Forni, S. (2012).** A common dataset for genomic analysis of livestock populations. *G3 : Genes/Genomes/Genetics* 2(4) :429–435. DOI 10.1534/g3.111.001453. — Le panel de porc PIC.

Annexe : la taille du support — un encadrement

Le coût de support-first suit la taille du support $|S|$; on l’encadre (preuves complètes, identité KKT de liaison et carte empirique dans la note compagne `research/support_bound_sketch.md`, scripts reproductibles `research/bound_*.py`).

Théorème 1 (existence sans ridge ni plafonds effectifs). Pour $\varepsilon = 0$, $s > 0$, faisabilité non vide et budgets simplex/sexués bornés, il existe un optimum avec $|S| \leq \text{rang}([\mathbf{A}; \mathbf{Z}^\top; \mathbf{b}^\top]) \leq q+r+1$ sans plafonds effectifs. Figer l'image $\mathbf{Z}^\top \mathbf{c}$ et l'objectif $\mathbf{b}^\top \mathbf{c}$ d'un optimum atteint définit une tranche bornée entièrement optimale. Un sommet ayant plus de coordonnées positives que le rang empilé admettrait deux perturbations opposées du noyau. Aucune dualité n'est requise; ce n'est pas une garantie de sortie de l'implémentation. Avec plafonds, un sommet satisfait seulement $|F| \leq q+r+1$ et $|S| = |F| + |U_+|$. $u_i = 2/n$, $\mathbf{G} = 0$, $q = 1$, $r = 0$ imposent déjà $|S| \geq \lceil n/2 \rceil$.

Théorème 2 (aucune borne universelle avec ridge). Pour $\varepsilon > 0$, aucune borne totale $f(q, r)$ indépendante de n n'existe. Sur le simplexe sans plafonds effectifs, $\mathbf{G} = \varepsilon \mathbf{I}$ donne une parenté uniforme ε/n et une parenté au moins $\varepsilon/(n-1)$ à tout plan avec un zéro. Ainsi $\varepsilon/n < k < \varepsilon/(n-1)$ impose le support plein à tout optimum réalisable. L'ancien balayage $n = 300$ est un contexte empirique; l'intervalle rigoureux et le témoin plafonné sont vérifiés séparément dans `bound_validation.py`.

L'identité KKT conditionnelle exige $\varepsilon > 0$, des multiplicateurs existants et $\lambda > 0$. Elle vaut seulement sur les coordonnées libres, inclut les multiplicateurs des deux bornes et utilise $m + q + 1$ coefficients. Le passage $\varepsilon \rightarrow 0$ ne fournit aucun théorème de support. Une parenté redondante permet un sommet LP non plafonné de support $\leq \text{rang}(\mathbf{A})$; la seule non-saturation en un optimum ne suffit pas.

Les supports opérationnels observés vont de quelques dizaines à quelques centaines (souris 47/89/182 au proxy $\Delta F_{\text{proxy}} = 2/1/0,5\%$); les 15–30 historiques provenaient de caps lâches. Une borne sous hypothèses spécifiées de spectre, critère et cap reste ouverte. Développement : `research/support_bound_fr.md` et `support_bound_fr.tex`.