# Sycomore ++ , un registre distribué orienté graphe auto-adaptatif

Aimen Djari, Emmanuelle Anceaume, Sara Tucci-Piergiovanni

▶ **To cite this version:**

Aimen Djari, Emmanuelle Anceaume, Sara Tucci-Piergiovanni. Sycomore ++ , un registre distribué orienté graphe auto-adaptatif. AlgoTel 2022 - 24èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, May 2022, Saint-Rémy-Lès-Chevreuse, France. pp.1-4. hal-03656546

**HAL Id: hal-03656546**
**https://hal.science/hal-03656546**

Submitted on 2 May 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

# *Sycomore$^{++}$, un registre distribué orienté graphe auto-adaptatif*

Aimen Djari[1], Emmanuelle Anceaume[2], Sara Tucci-Piergiovanni[1]

[1] *University Paris-Saclay, CEA, List, Palaiseau, France*
[2] *CNRS/IRISA, France*

L'arrivée de Bitcoin a entraîné le passage vers des écosystèmes décentralisés grâce à l'échange de transactions sans tiers de confiance. Cependant, parmi les principaux défis auxquels doivent faire face les blockchains non permissionnées figurent la scalabilité et la sécurité. C'est dans cette optique qu'a été créé Sycomore, un registre distribué orienté graphe dont la structure s'auto-adapte au débit de transactions soumises. Dans cet article, nous présentons et évaluons une extension de Sycomore, Sycomore$^{++}$ dont la principale caractéristique est l'auto-adaptation de son débit de création de blocs au débit de transactions soumises. Nous comparons ensuite ses performances à celles de Bitcoin et de Sycomore. En s'appuyant sur des simulations par agents, nous évaluons la capacité de ces registres distribués à relever les défis susmentionnés, dans différents contextes d'exécution. L'un des principaux enseignements tirés de ces simulations est la capacité de Sycomore$^{++}$ à réduire considérablement le temps de confirmation des transactions, à réagir rapidement à toute variation soudaine du débit de soumission des transactions et à minimiser le gaspillage d'énergie, en comparaison avec les autres registres distribués basés sur le Proof-of-Work précédemment cités.

**Mots-clefs :** Registre distribué, graphe, scalabilité, simulation basée agents.

## 1 Introduction

Afin de résoudre les problèmes de performance des registres distribués non permissionnés de type blockchain (chaîne de blocs), et en particulier les problèmes de scalabilité tels que le faible débit de confirmation des transactions, de nouvelles conceptions ont été proposées, notamment BITCOIN-NG [6], qui privilégie un mécanisme hors-chaîne dans lequel un leader est chargé de valider les transactions en dehors de la blockchain ; LIGHTNING [9], qui suit le même principe mais ne publie que le résultat des transactions successives entre un ensemble de parties. D'autres propositions telles que HashGraph [2] ou Iota [3] envisagent de se débarrasser des blocs, tandis que la famille de protocoles Ghost [11] et Spectre [10] modifie la structure de la blockchain, qui passe d'une séquence totalement ordonnée de blocs à un graphe dirigé de blocs. Les blocs sont construits de telle sorte qu'ils valident l'état du graphe au moment où les blocs sont créés, ce qui réduit la possibilité pour les attaquants de créer des blocs à l'avance. En ce qui concerne l'approche orientée graphe, l'absence de mécanismes pour empêcher la présence de conflits (c'est-à-dire de blocs avec des transactions conflictuelles) ou la présence de boucles dans le graphe dirigé (Spectre [10] organise les blocs dans un graphe dirigé, mais non acyclique, de blocs) exige que les participants exécutent un algorithme complexe pour extraire du graphe l'ensemble des transactions acceptées (c'est-à-dire valides).

**Sycomore.** Sycomore est un registre distribué non permissionné et immuable dont la structure est un graphe dirigé acyclique, appelé SYC-DAG [1]. Sa conception diffère des registres distribués existants sur la structure de son graphe qui s'adapte dynamiquement aux fluctuations du débit de soumission des transactions : Lorsque le bloc feuille d'une chaîne (plus précisément le dernier bloc ajouté à une chaîne) du graphe dépasse un seuil de charge maximal (la charge est mesurée en octets), les blocs suivants sont répartis sur deux chaînes *siblings* (sœurs), et ces blocs sont minés en parallèle (par deux mineurs différents). Inversement, lorsque les blocs feuilles de deux chaînes *siblings* sont inférieurs à un seuil de charge minimal, les blocs suivants appartiendront à une chaîne unique. La décision de *split* (fractionner) une chaîne du SYC-DAG ou de *merge* (fusionner) deux chaînes *siblings* est prise localement par chaque mineur, et le

"bien-fondé" de cette décision est vérifiable par tous à tout moment. Sycomore a été conçu pour satisfaire les propriétés suivantes [1] : **(P1) Auto-adaptation à la charge des transactions.** L'augmentation et la diminution du débit de soumission des transactions sont gérées dynamiquement par la création ou la disparition progressive de chaînes dans le SYC-DAG ; **(P2) Partitionnement équilibré des transactions.** Il n'existe aucune transaction qui appartienne à deux blocs différents ; **(P3) Imprévisibilité du prédécesseur.** La chaîne à laquelle un nouveau bloc est annexé ne peut être ni choisie ni prédite ; **(P4) Équité de la chaîne.** Toutes les chaînes du SYC-DAG croissent à la même vitesse ; **(P5) Probabilité de forks négligeable.** La probabilité de forks est maximale lorsque le SYC-DAG est réduit à une seule chaîne (c'est-à-dire $1,2\times10^{-3}$ dans l'intervalle de temps de 30 secondes) et diminue proportionnellement au nombre de blocs feuilles.

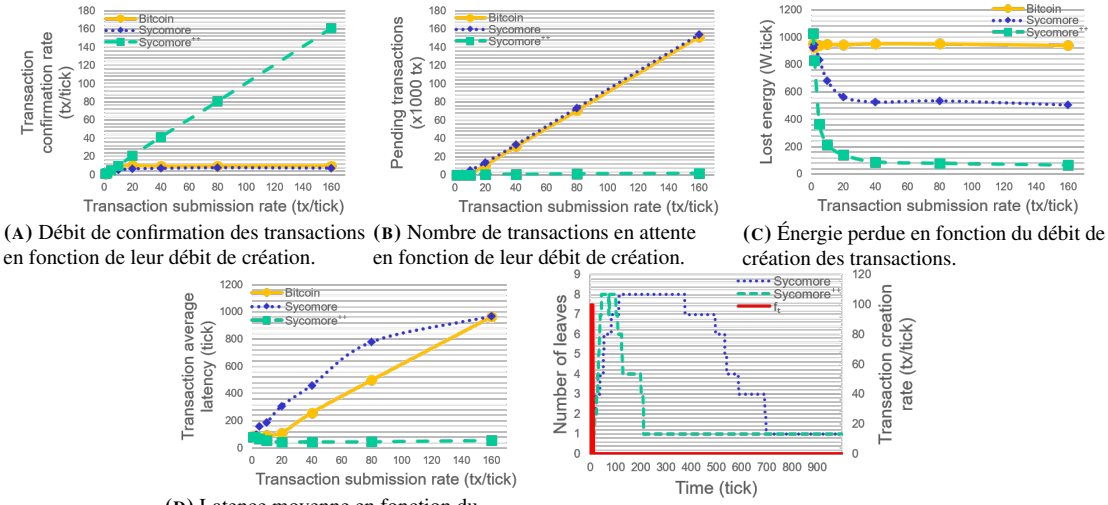
**Motivations.** En adaptant dynamiquement la largeur du graphe, c'est-à-dire le nombre de chaînes du SYC-DAG, à la charge de transaction réelle du réseau, on pourrait s'attendre à ce que Sycomore garantisse une latence de transaction presque optimale. Par latence de transaction, on entend le temps qui s'écoule entre l'instant où un utilisateur soumet une transaction au réseau et l'instant où cette transaction est confirmée dans le SYC-DAG, c'est-à-dire qu'elle appartient à un bloc qui est profondément installé dans le SYC-DAG. Ceci est vrai juste après le réajustement périodique de la difficulté de minage. En effet, dans la plupart des registres distribués basés sur la Proof-of-Work (preuve de travail), la difficulté de minage est périodiquement calibrée pour faire face aux variations de la puissance de calcul du réseau. Dans Sycomore, un tel réajustement est exécuté chaque fois que la hauteur du SYC-DAG augmente de $H_{max}$ blocs depuis la dernière période de réajustement, où $H_{max} = 2016$ comme dans Bitcoin (c'est-à-dire tous les 14 jours). Cependant, il est fort probable que pendant une période de 14 jours, le taux de soumission des transactions varie radicalement, modifiant en conséquence la largeur du SYC-DAG. Par conséquent, la difficulté de minage qui a été calculée pour s'adapter à la fois à la puissance de calcul du réseau et à l'état du graphe lors du dernier réajustement peut actuellement être soit sous-estimée soit surestimée par rapport à l'état actuel du SYC-DAG. Bien que ceci s'applique à d'autres registres distribués basés sur le PoW, les conséquences dans Sycomore peuvent être amplifiées par rapport aux autres protocoles en raison de son adaptation dynamique. Considérons un SYC-DAG d'une très grande largeur (c'est-à-dire qu'un grand nombre de blocs valides peuvent être ajoutés en parallèle au SYC-DAG) au moment du réajustement. Dans le scénario où le nombre de transactions soumises chute de manière significative, cela entraînerait une diminution progressive du nombre de chaînes, et donc une difficulté de minage sous-estimée par rapport à l'état actuel du SYC-DAG. Un tel scénario montre une possible brèche de sécurité : Les mineurs adversaires peuvent profiter d'une difficulté de minage trop faible pour dégrader la qualité du registre [7], c'est-à-dire la proportion maximale de blocs apportés par l'adversaire dans le registre de tout nœud honnête. Le scénario inverse peut également se produire, où une augmentation soudaine du taux de transaction donnera lieu à une difficulté de minage surestimée, et donc à une consommation d'énergie très élevée, qui sera dans le pire des cas similaire à celle de Bitcoin. En conséquence, la latence des transactions sera très élevée jusqu'au prochain réajustement.

## 2 Sycomore$^{++}$

Pour prévenir ces problèmes critiques, nous proposons une extension de Sycomore, appelée Sycomore$^{++}$, qui vise à garantir un délai inter-blocs constant sur toutes les chaînes quelle que soit la structure du SYC-DAG [5]. L'idée principale de Sycomore$^{++}$ est d'auto-adapter la difficulté de minage au nombre de chaînes du SYC-DAG, c'est-à-dire de diviser la difficulté de minage par le nombre actuel de chaînes. Notons que cet ajustement adaptatif ne remplace pas le réajustement périodique. Le premier adapte la difficulté à la structure du SYC-DAG tandis que le second adapte la difficulté à la puissance de calcul globale du système.

**Validation expérimentale** Pour valider le comportement de Sycomore$^{++}$ et pour le comparer à Sycomore et à Bitcoin, nous avons implémenté [†] ces trois protocoles distribués sur un simulateur à base d'agents dédié aux systèmes blockchain, appelé Multi-Agent eXperimenter (MAX) [8]. MAX offre des bibliothèques génériques permettant de développer des protocoles de registres distribués et un large éventail de scénarios de simulation. Il s'agit d'un simulateur d'événements discrets, où l'unité de temps de simulation est appelée *tick*. Les primitives de communication nous permettent de configurer différents modèles de communication.

†. Les codes sources de Bitcoin, Sycomore et Sycomore$^{++}$ ainsi que tous les scripts de nos expérimentations sont accessibles [12]




**(A)** Débit de confirmation des transactions en fonction de leur débit de création.

**(B)** Nombre de transactions en attente en fonction de leur débit de création.

**(C)** Énergie perdue en fonction du débit de création des transactions.

**(D)** Latence moyenne en fonction du débit de création des transactions.

**(E)** Réactivité de Sycomore et de Sycomore$^{++}$ en présence d'un pic de transactions.

**FIGURE 1 :** Scalabilité de Bitcoin, Sycomore et Sycomore$^{++}$ et Réactivité de Sycomore et Sycomore$^{++}$.

Pour cette étude, le modèle configuré est une diffusion fiable des messages avec un délai configurable ‡. Dans nos simulations, un *tick* équivaut à une minute ; la difficulté de minage est calibrée de façon à avoir un bloc tous les 10 ticks ; le nombre de txs (transactions) par bloc est fixé à 100 (contre 4000 en réalité), et ce pour alléger les simulations. Toutes les expérimentations sur Sycomore$^{++}$, Sycomore et Bitcoin ont été réalisées sur Grid'5000 [4], un banc d'essai grande échelle utilisé ici pour paralléliser l'exécution de plusieurs scénarios de simulation (variation de paramètres). Dans cet article, nous nous concentrons uniquement sur les propriétés de scalabilité et de réactivité. La figure 1 illustre les principaux résultats concernant la capacité de Sycomore$^{++}$, Sycomore et Bitcoin à gérer des débits élevés de soumission de transactions. Plus précisément, nous évaluons le débit de confirmation des transactions, la latence (c'est-à-dire le temps moyen écoulé entre la soumission d'une transaction et sa confirmation), ainsi que le nombre moyen de transactions non confirmées (c'est-à-dire toujours en attente d'être intégrées dans un bloc en fin de simulation) en fonction du débit de soumission des transactions $f_t$ (nombre de transactions soumises par tick), ce dernier étant fixé au début de chaque expérimentation. L'énergie perdue (c'est-a-dire la somme, pour chaque mineur $u$ et pour chaque bloc créé $b$ non ajouté au SYC-DAG, du nombre de ticks passés à travailler sur $b$ multiplié par la puissance de calcul $cp_u$) par les mineurs lors de la création des blocs est également mesurée.

Intéressons-nous d'abord au débit de confirmation des transactions en fonction de leur débit de création (voir Figure 1a). La principale observation concernant Bitcoin et Sycomore est que, quelle que soit la puissance de calcul du réseau, pas plus de 10 transactions/tick ne sont confirmées, ce qui illustre l'impact du délai inter-bloc globalement constant (i.e. un bloc est miné tous les 10 ticks en moyenne). Sycomore présente des résultats légèrement inférieurs à ceux de Bitcoin, ce qui est dû à l'augmentation du nombre de chaînes du SYC-DAG, dans lesquelles les blocs peuvent être moins chargés. D'autre part, en adaptant continuellement la difficulté de minage au nombre de blocs feuilles, et donc à $f_t$, Sycomore$^{++}$ présente un comportement optimal en ce qui concerne la confirmation des transactions, c'est-à-dire que pour tout $f_t$, le débit de confirmation des transactions est égal au débit de création des transactions. À titre de comparaison, 160 txs/tick dans notre scénario de simulation (en réalité, nous ne pouvons pas insister davantage sur le simulateur par manque de puissance) correspond a 6400 txs/mn dans la vie réelle. La figure 1b montre le nombre moyen de transactions non confirmées, c'est-à-dire le nombre moyen de transactions qui se sont accumulées chez les mineurs sans avoir pu être intégrées dans des blocs. Pour Bitcoin et Sycomore, ce nombre augmente linéairement avec le débit de soumission des transactions une fois qu'il dépasse 10txs/tick puisque cela correspond au délai global de création inter-blocs. En revanche, en adaptant le nombre de blocs créés à $f_t$, Sycomore$^{++}$ réduit drastiquement le nombre moyen de transactions non confirmées. Par exemple, pour $f_t = 10$ txs/tick, ce nombre est égal à 432 transactions, et pour $f_t = 160$ txs/tick, il est égal à $1,957$ transac-

‡. Dans ce résumé, nous considérons ce délai comme nul. L'analyse du cas général se trouve dans [5].

tions, comparé à 154, 000 dans Bitcoin et Sycomore. La figure 1c illustre l'énergie perdue en fonction de $f_t$. Dans Bitcoin, comme le délai inter-blocs, le nombre de mineurs et la difficulté ne varient pas, la même quantité d'énergie est perdue indépendamment de $f_t$. Bien que ce paramètre s'applique également à Sycomore, le fait que le SYC-DAG devienne plus large avec des valeurs croissantes de $f_t$ donne lieu à une distribution uniforme de la puissance de calcul sur les feuilles du SYC-DAG, et diminue ainsi la concurrence entre mineurs. Ainsi, moins d'énergie gaspillée par rapport à Bitcoin. En ce qui concerne Sycomore$^{++}$, pour des valeurs croissantes de $f_t$, le SYC-DAG devient plus large et les blocs sont créés plus rapidement (puisque la difficulté de minage s'adapte à la structure du SYC-DAG), ce qui permet à Sycomore$^{++}$ d'atteindre un nombre optimal de chaînes plus rapidement que Sycomore. En conséquence, nous obtenons une meilleure parallélisation du travail des mineurs, et donc une réduction drastique de la perte d'énergie. La figure 1d illustre la latence moyenne des transactions en fonction de $f_t$. Rappelons que la latence des transactions mesure le temps écoulé entre l'instant où une transaction est envoyée au réseau par l'utilisateur et celui où elle est confirmée. Contrairement à toutes les autres expérimentations, la latence des transactions a été mesurée comme suit : les transactions ne sont envoyées que pendant un certain temps à $f_t$, et les simulations s'arrêtent une fois que toutes les transactions soumises ont été confirmées. La première observation est que, dans le cas de Bitcoin, une fois que $f_t \geq 10$ txs/tick, la latence des transactions augmente linéairement avec $f_t$, ce qui corrobore clairement les figures 1a et 1b. En ce qui concerne Sycomore, la perte de performance par rapport à Bitcoin est due au fait que les blocs ne sont pas nécessairement entièrement remplis, ce qui retarde d'autant la latence des transactions. En revanche, Sycomore$^{++}$ bénéficie d'une latence moyenne constante (égale à 50 ticks quelle que soit la valeur de $f_t$). En fait, plus il y a de transactions envoyées, plus les blocs sont remplis, et ce, jusqu'à ce que la forme optimale du graphe soit atteinte (ce qui peut prendre du temps sur Sycomore), ce qui fait que les premiers blocs sont souvent chargés à 100%. Ce phénomène est notamment perceptible pour $f_t = 160$ txs/tick.
Enfin, la figure 1e illustre la réactivité de Sycomore et de Sycomore$^{++}$ (nous omettons Bitcoin de cette évaluation puisque sa structure à une chaîne ne s'adapte pas au débit de transactions), c'est-à-dire leur capacité à réagir aux fluctuations soudaines de $f_t$, illustrées par la présence d'un pic de charge (représenté par la fonction constante rouge de $t = 1$ à $t = 11$ ticks à $f_t = 100$txs/tick). Sycomore et Sycomore$^{++}$ subissent initialement une série de *split*, puis passent progressivement à une série de *merge* jusqu'à converger vers une structure à une chaîne. Sycomore$^{++}$ diffère de Sycomore par sa rapidité à *split* et à *merge* : Sycomore$^{++}$ réussit à faire face à la hausse de débit 75% plus vite que Sycomore, et à sa diminution 33% plus vite.

## 3 Travaux en cours et futurs

Dans cet article, nous avons présenté Sycomore$^{++}$, un registre distribué orienté graphe qui tire clairement avantage de la puissance de calcul fournie par le système, et réussit à toujours garantir un débit de confirmation des transactions qui s'adapte à leur débit de création. Parmi les travaux futurs, nous prévoyons de continuer l'étude de certains paramètres de Sycomore tel que $H_{max}$. Nous envisageons également un possible remplacement de la Proof-of-Work par un autre type de consensus (Proof-of-Stake par exemple).

## Références


[1] E. Anceaume, A. Guellier, R. Ludinard, and B. Sericola. Sycomore : A permissionless distributed ledger that self-adapts to transactions demand. In *NCA*, 2018.

[2] L. Baird. The swirlds hashgraph consensus algorithm : Fair, fast, byzantine fault tolerance, TR. 2016.

[3] G. Bu, Ö. Gürcan, and M. Potop-Butucaru. G-IOTA : fair and confidence aware tangle. *CoRR*, 2019.

[4] D. Balouek, et al. Adding virtualization capabilities to the Grid'5000 testbed. In *Cloud Computing and Services Science*. 2013.

[5] A. Djari, E. Anceaume, and S. Tucci-Piergiovanni. An agent-based simulation study of Sycomore ++ , a scalable and self-adapting graph-based permissionless distributed ledger. under submission, February 2022.

[6] I. Eyal, A. E. Gencer, E. Gün Sirer, and R. Van Renesse. Bitcoin-NG : A scalable blockchain protocol. In *NSDI*, 2016.

[7] J. A. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol : Analysis and Applications. In *EUROCRYPT*, 2015.

[8] N. Lagaillardie, A. Djari, and Ö. Gürcan. A computational study on fairness of the tendermint blockchain protocol. *MDPI*, 2019.

[9] J. Poon and T. Dryja. *The bitcoin lightning network*. 2016.

[10] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. Spectre : A fast and scalable cryptocurrency protocol. *IACR*, 2016.

[11] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR*, 2013.

[12] Sycomore$^{++}$. Source code. `https://anonymous.4open.science/r/Sycomorepp-412D`.