Titre: Title:	Vers une mise à l'échelle efficace des protocoles de prêt et d'emprunt en pair à pair sur Ethereum
Auteur: Author:	Julien Thomas
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Thomas, J. (2023). Vers une mise à l'échelle efficace des protocoles de prêt et d'emprunt en pair à pair sur Ethereum [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/57013/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/57013/
Directeurs de recherche: Advisors:	Foutse Khomh, & Marios-Eleftherios Fokaefs
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Vers une Mise à l'Échelle Efficace des Protocoles de Prêt et d'Emprunt en Pair à Pair sur Ethereum

JULIEN THOMAS
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Décembre 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Vers une Mise à l'Échelle Efficace des Protocoles de Prêt et d'Emprunt en Pair à Pair sur Ethereum

présenté par **Julien THOMAS**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Jinghui CHENG, président
Foutse KHOMH, membre et directeur de recherche
Marios-Eleftherios FOKAEFS, membre et codirecteur de recherche
Catherine BEAUDRY, membre

DÉDICACE

À tous mes amis et collègues,
de chez Morpho Labs...

REMERCIEMENTS

Je tiens à exprimer ma profonde gratitude à Merlin, Mathis et Paul, mes co-fondateurs chez Morpho Labs. Depuis plus de deux ans, nous partageons ensemble cette passionnante aventure dans le monde fascinant de la finance décentralisée. Leur esprit d'équipe, leur soutien et leur créativité ont été des piliers essentiels de notre succès commun.

Un remerciement tout particulier à l'ensemble de l'équipe de Morpho Labs. Chaque jour, leur engagement, leur enthousiasme et leur dévouement dans le développement de solutions novatrices contribuent à façonner l'avenir de la finance. Leur soutien a été une source d'inspiration constante et a grandement enrichi mon expérience professionnelle et personnelle.

Je tiens également à exprimer ma profonde reconnaissance à ma famille. Un merci spécial à mes sœurs, Laura et Océane pour leur soutien indéfectible et leur présence rassurante à chaque étape de mon parcours.

RÉSUMÉ

L'avènement de la blockchain Ethereum a ouvert la voie à une nouvelle ère de solutions financières décentralisées, communément appelées finance décentralisée (DeFi). Parmi les innovations majeures dans ce domaine, les protocoles de prêt et d'emprunt ont émergé comme des composants essentiels, offrant des mécanismes de crédit décentralisés sans les intermédiaires traditionnels. Ces protocoles permettent aux utilisateurs de prêter leurs actifs cryptographiques ou d'emprunter contre des garanties de manière autonome et sécurisée sur la blockchain.

Ce mémoire se concentre sur l'analyse de la mise à l'échelle de ces protocoles de prêt et d'emprunt en pair à pair sur Ethereum. Il vise à explorer les défis, les solutions, et les optimisations possibles pour améliorer l'efficacité et l'accessibilité de ces systèmes dans un environnement en constante évolution.

Nous débuterons par une exploration des fondements des protocoles de prêt et d'emprunt en DeFi, en mettant l'accent sur des plateformes populaires telles que Aave et Compound. Nous analyserons ensuite comment ces systèmes peuvent être optimisés pour gérer plus efficacement la liquidité et les risques, tout en assurant une évolutivité accrue sur la blockchain Ethereum.

Un élément central de notre étude sera l'examen de Morpho, une surcouche innovante conçue pour améliorer l'efficacité des protocoles de prêt existants. Nous étudierons comment Morpho facilite des interactions plus directes entre prêteurs et emprunteurs, réduisant ainsi la dépendance aux pools de liquidité traditionnelles et favorisant une meilleure utilisation des ressources financières disponibles.

Ensuite, nous concentrerons sur le développement et la validation d'un mécanisme innovant, le Delta, permettant d'améliorer la mise à l'échelle de Morpho afin d'optimiser son efficacité et sa mise à l'échelle dans l'écosystème de la finance décentralisée.

Enfin, nous discuterons des implications de ces technologies pour le futur de la finance décentralisée, en soulignant les opportunités et les défis que représentent la mise à l'échelle et l'optimisation des protocoles de prêt et d'emprunt en pair à pair sur Ethereum. Cette analyse sera complétée par une réflexion sur les perspectives futures et les développements potentiels dans ce domaine en pleine expansion.

ABSTRACT

The Ethereum blockchain has ushered in a new era of decentralized financial solutions, known as decentralized finance (DeFi). Among the significant innovations in this area, lending and borrowing protocols have emerged as essential components, offering decentralized credit mechanisms without traditional intermediaries. These protocols allow users to lend their crypto assets or borrow against collateral autonomously and securely on the blockchain.

This thesis analyzes the scaling of these peer-to-peer lending and borrowing protocols on Ethereum. It aims to explore challenges, solutions, and possible optimizations to improve the efficiency and accessibility of these systems in a constantly evolving environment.

We'll begin by exploring the foundations of lending and borrowing protocols in DeFi, focusing on popular platforms such as Aave and Compound. We will then analyze how these systems can be optimized to manage liquidity and risk more effectively while ensuring increased scalability on the Ethereum blockchain.

Next, we will devise mechanisms aimed at scaling Morpho on Ethereum. Morpho is an innovative overlay designed to improve the efficiency of existing lending protocols. We will also explore how Morpho facilitates more direct interactions between lenders and borrowers, reducing reliance on traditional liquidity pools and promoting better use of available financial resources.

Finally, we will discuss the implications of these technologies for the future of decentralized finance, highlighting the opportunities and challenges of scaling and optimizing peer-to-peer lending and borrowing protocols on Ethereum. This analysis will be supplemented by a reflection on future perspectives and potential developments in this rapidly expanding field.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

PLF	Protocol for Loanable Funds
DeFi	Decentralized Finance
TVL	Total Value Locked
EVM	Ethereum Virtual Machine
MEV	Maximum Extractable Value
EF	Fondation Ethereum
EIP	Ethereum Improvements Proposals
ERC	Ethereum Requests for Comments
EOA	Externally Owned Address

Nomenclature

U	L'utilisation d'une pool
U_{opt}	L'utilisation optimale d'une pool
$\bar{r}^S$	Taux moyen correspondant à l'indice λ^S entre les instants t et $t-1$
δ^B	Delta d'emprunt sur Morpho
δ^S	Delta de prêt sur Morpho
λ^{γ^B}	Indice d'emprunt pair à pair avec Delta
λ^{γ^S}	Indice de prêt pair à pair avec Delta
λ^{ρ^B}	Indice d'emprunt pair à pair avec un facteur de réserve
λ^{ρ^S}	Indice de prêt pair à pair avec un facteur de réserve
λ_θ^{P2P}	Indice pair à pair médian (sans delta et sans frais)
$\omega_\theta^B(u)$	Balance d'emprunt d'un utilisateur u sur Morpho
$\omega_\theta^S(u)$	Balance de prêt d'un utilisateur u sur Morpho
$b_\theta(\overline{M_\psi})$	Position d'emprunt de Morpho ($\overline{M_\psi}$) sur la pool Θ
B_a	La somme des emprunts sur la pool a
b_u	L'emprunt (borrow) d'un utilisateur u
b_u^B	Balance d'emprunt d'un utilisateur u
b_u^S	Balance de prêt d'un utilisateur u
e	Efficacité de l'algorithme de Mariage
lf_a	Le facteur de liquidation associé à un marché a
N	Borne maximale du nombre de pairage par transaction dans l'algorithme de Mariage.
r^α	Le taux en pair à pair, sans delta ni frais

r^{γ^B} Taux d'emprunt pair à pair avec Delta

r^{γ^S} Taux de prêt pair à pair avec Delta

r^{ρ^B} Le taux en pair à pair des emprunteurs, avec frais et sans delta

r^{ρ^S} Le taux en pair à pair des prêteurs, avec frais et sans delta

r^B_{linear} Le taux (rate) d'emprunt linéraire

r^S_{linear} Le taux (rate) de prêt linéraire

$s_\theta(\overline{M_\psi})$ Position de prêt de Morpho ($\overline{M_\psi}$) sur la pool Θ

S_a La somme des dépots sur la pool a

s_u Le prêt (supply) d'un utilisateur u

U_a L'ensemble des utilisateurs de la pool a

λ^B_θ Indice sur la pool pour les emprunteurs (Borrowers)

λ^S_θ Indice sur la pool pour les prêteurs (Suppliers)

CHAPITRE 1 INTRODUCTION

1.1 Contexte

Les chaînes de blocs, ou *blockchains*, ont révolutionné le monde numérique en offrant une approche innovante de la gestion des données et des transactions. À l'origine développées comme la technologie sous-jacente du Bitcoin, elles ont rapidement dépassé le cadre des crypto-monnaies pour s'imposer comme une solution majeure dans divers secteurs. Aujourd'hui, elles représentent bien plus qu'une simple méthode de transaction financière : elles sont au cœur d'une transformation numérique globale. Cette évolution est due à leur capacité unique de fournir un registre décentralisé, transparent et sécurisé, permettant ainsi des interactions sans intermédiaire de confiance.

Dans ce mémoire, nous explorons les applications pratiques et les défis inhérents à ces technologies, en mettant l'accent sur les protocoles de prêt et d'emprunt dans la finance décentralisée.

L'étude de ces protocoles sur la blockchain Ethereum, notamment leur mise à l'échelle et leur efficacité, est essentielle pour comprendre comment ils peuvent remodeler le paysage financier et ouvrir la voie à de nouvelles formes d'interactions économiques dans le monde numérique.

1.2 Définitions et concepts de base

Dans cette partie introductive, nous allons principalement nous pencher sur les fondements d'Ethereum [1] et de son application à la Finance Décentralisée. Cette compréhension est primordiale pour la suite, qui va largement aborder les protocoles de prêt et d'emprunts, tous développés sur Ethereum. Nous verrons également que développer sur Ethereum diffère considérablement du développement logiciel classique, notamment en raison de l'introduction d'un coût de Gas lié à l'utilisation.

1.2.1 La finance décentralisée

Les chaînes de blocs publiques comme Ethereum [1] sont aujourd'hui principalement utilisées à travers des protocoles financiers dans un domaine appelé la finance décentralisée. Ce secteur regroupe notamment de multiples monnaies et permet de les échanger, de les prêter, de les emprunter ou même de les conserver sans intermédiaire de confiance, uniquement grâce à des contrats intelligents qui assurent le fonctionnement de cet écosystème de manière autonome

[2].

Définition 1.2.1.1. *On appelle TVL (de l'anglais Total Value Locked) la liquidité normalisée, généralement exprimée en dollars, qui est disponible sur un protocole donné. Dans le cas particulier des protocoles de prêts et d'emprunts, il s'agit du montant total des fonds déposés, soustrait des emprunts.*

La finance décentralisée offre un large éventail d'applications, comme effectuer des échanges de monnaies sans intermédiaires, définir des taux d'intérêt dynamiques ou optimiser automatiquement des stratégies de placement. On a observé une croissance significative de leur utilisation depuis début 2021 (voir Figure 1.1). Nous allons principalement nous concentrer sur les protocoles de prêt et d'emprunt.

Zoom sur les protocoles de prêts et d'emprunts

Ces protocoles ont rapidement émergé dans ce contexte et ont connu une adoption rapide. En 2020, plus de 40 milliards de dollars en liquidités étaient déposés dans ces protocoles (voir l'exemple d'Aave sur la Figure 1.2). Compound [3] et Aave [4] se sont particulièrement distingués grâce à leur simplicité d'utilisation et de compréhension.

Définition 1.2.1.2. *Le taux de collatéralisation représente la valeur supplémentaire qu'un utilisateur doit fournir pour emprunter un montant donné. Les protocoles de prêts et d'emprunts étudiés dans ce mémoire sont dits surcollatéralisés, c'est-à-dire que le taux de collatéralisation est supérieur à 100%.*

Toutefois, l'environnement d'exécution de la blockchain est restrictif en raison de la nature du système distribué. Les protocoles doivent donc faire des hypothèses et des choix de conception parfois au détriment de leur efficacité. Ainsi, les modèles choisis pour les prêts et emprunts sont des modèles en "pool", où les prêteurs sont regroupés et partagent les intérêts payés par les emprunteurs, créant ainsi un écart entre les taux d'intérêts des prêteurs et des emprunteurs.

FIGURE 1.1 TVL & volume en dollars de la finance décentralisée sur Ethereum

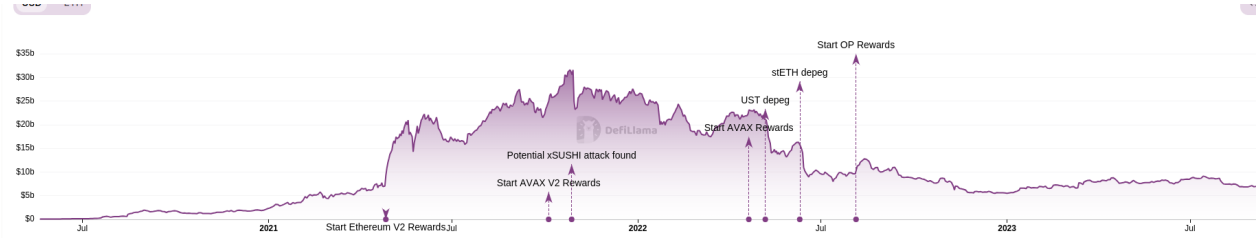


FIGURE 1.2 TVL & volume en dollars de Aave sur Ethereum

Il existe cependant des modèles mathématiquement plus performants qui pourraient améliorer l'efficacité des protocoles, mais leur mise en œuvre sur une blockchain n'ont jamais abouti, due a des problèmes de mise à l'échelle. Le meilleur exemple en date est EthLend [5], protocole en pari à pair qui a découlé sur Aave ensuite. L'échec de EthLend est en parti due a la difficulté de développer une bonne expérience utilisateur sur Ethereum.

1.2.2 Les défis du développement sur Ethereum

Le développement sur Ethereum présente des défis uniques. L'un des principaux défis est le coût du gas, une unité de mesure représentant la quantité de ressources de calcul et de stockage nécessaires pour effectuer des opérations. Chaque transaction sur Ethereum nécessite un certain montant de gas, et le prix du gas fluctue en fonction de la demande sur le réseau. Ceci impose une contrainte de coût aux utilisateurs et aux développeurs de protocoles, les incitant à optimiser leurs contrats intelligents pour minimiser l'utilisation du gas.

En outre, la nature immuable des contrats sur Ethereum signifie que, une fois déployés, ils ne peuvent être modifiés. Cela nécessite une attention particulière lors de la conception et du déploiement des contrats intelligents, car toute erreur ou faille de sécurité peut avoir des conséquences irréversibles.

1.2.3 Problématiques des protocoles de prêt et d'emprunt

Les protocoles de prêt et d'emprunt sur Ethereum, malgré leur popularité, font face à des défis significatifs en termes d'efficacité et de mise à l'échelle. Comme mentionné précédemment, ces protocoles fonctionnent principalement sur un modèle de pool, où les prêteurs mettent leurs actifs en commun. Bien que ce modèle facilite l'accès des emprunteurs à la liquidité, il crée un écart entre les taux d'intérêt perçus par les prêteurs et ceux payés par les emprunteurs.

Ce modèle de pool, tout en étant efficace pour rassembler la liquidité, peut ne pas être optimal pour maximiser le rendement des prêteurs ou minimiser les coûts pour les emprunteurs. De

plus, la gestion des risques dans ces pools centralisées peut être complexe, nécessitant des mécanismes de gouvernance et de sécurité robustes.

1.3 Éléments de la problématique

Dans le paysage évolutif de la finance décentralisée, Morpho émerge comme une innovation significative. Décrit dans son livre blanc [6], Morpho s'appuie sur le modèle de pool existant et y superpose une couche de pair à pair pour associer directement les prêteurs aux emprunteurs, assurant ainsi une uniformité des taux d'intérêt pour les deux parties en pair à pair. Lorsqu'un appariement direct n'est pas possible, Morpho se replie sur les pools de liquidité préexistants sur Ethereum. En adoptant un taux intermédiaire entre les taux de prêt et d'emprunt des pools, Morpho garantit à ses utilisateurs un taux égal ou supérieur à celui des pools. Ce système incarne une amélioration de Pareto : il offre les mêmes risques et liquidités tout en améliorant les taux d'intérêt.

Toutefois, la stratégie de mise en pair à pair décrite dans le livre blanc de Morpho [6], en particulier dans la section 2.4 dédiée au "matching engine", se révèle coûteuse. La complexité linéaire de cette opération par rapport au nombre d'utilisateurs à apparier rend son coût prohibitif sur Ethereum, compte tenu de la structure de frais inhérente à cette plateforme.

Il est donc impératif de concevoir une stratégie d'optimisation pour l'opération de pairage de Morpho qui soit non seulement compatible avec les contraintes d'Ethereum, mais qui exploite également au maximum les ressources offertes par Morpho. Un tel ajustement permettrait à Morpho de s'établir comme le premier protocole de prêt et d'emprunt en pair à pair sur une blockchain publique, optimisant l'efficacité sur Ethereum tout en préservant ses fonctionnalités essentielles et en maintenant son statut d'amélioration de Pareto.

1.4 Objectifs de recherche

Dans le contexte des contraintes inhérentes à Morpho et aux environnements de blockchain tels qu'Ethereum, notre recherche vise à développer un mécanisme permettant d'assurer l'efficacité et la fiabilité de Morpho, quel que soit le volume de transactions qu'il doit traiter.

L'objectif principal est de concevoir une solution applicable sur Ethereum, une plateforme qui présente des défis uniques en termes d'exécution de contrats intelligents, afin de garantir le bon fonctionnement de Morpho sur cette blockchain.

Nous définissons trois objectifs spécifiques pour ce projet de recherche :

— **Analyse approfondie de Morpho :** Cela comprend l'identification, l'analyse et

l'explication des mécanismes sous-jacents de Morpho, ainsi que de ses limitations en matière de mise à l'échelle. Nous envisageons de formaliser le protocole Morpho à travers des définitions rigoureuses, des propriétés et des théorèmes, et de développer un modèle de simulation pour observer les scénarios dans lesquels Morpho atteint ses limites opérationnelles en raison de contraintes de mise en pair à pair.

— **Développement d'un mécanisme de mise à l'échelle :** Nous cherchons à concevoir et implémenter une solution d'ingénierie pour la mise à l'échelle de Morpho sur Ethereum appelée "Delta". Cela inclura la validation des propriétés et des théorèmes du protocole Morpho dans ce nouveau contexte, en s'assurant que la complexité et le coût en gas du protocole restent gérables. Des lignes directrices spécifiques pour l'implémentation sur Ethereum seront également élaborées.

— **Évaluation comparative de Morpho avec Delta :** Enfin, nous utiliserons un outil de simulation pour évaluer la nouvelle version de Morpho, en intégrant le concept de Delta, et comparerons son efficacité avec celle des protocoles existants de prêt et d'emprunt. L'objectif est de mesurer la performance de Morpho en termes de coût d'utilisation, de mise à l'échelle et d'efficacité par rapport à des protocoles établis comme AaveV2.

Ces objectifs visent à surmonter les obstacles techniques actuels et à placer Morpho en tant que solution de prêt et d'emprunt à la pointe de la technologie blockchain, tout en respectant les contraintes strictes d'Ethereum.

1.5 Hypothèses et Assomptions de Recherche

La nature autonome et décentralisée des protocoles sur les blockchains signifie que des plates-formes comme Morpho sont conçues pour fonctionner sur le long terme, potentiellement sur des décennies. Cela introduit une difficulté considérable dans la prédiction des types et des comportements des utilisateurs futurs. Ainsi, nos simulations et analyses sont fondées sur un ensemble de données issues des transactions effectuées sur le marché USDC-AaveV2 sur Ethereum, entre les blocs spécifiés 11362579 (déploiement du marché USDC sur AaveV2) et 12000000. Cet échantillonnage nous permet d'aligner nos hypothèses avec les comportements actuels des utilisateurs sur les protocoles de prêt et d'emprunt, qui constituent la cible initiale des utilisateurs de Morpho.

Pour cette étude, nous utilisons ces transactions extraites de Aave, que nous réutiliserons afin de les faire passer dans des simulations de Morpho sous différentes formes.

Il est important de noter que Morpho comprend plusieurs axes de recherche, dont la structure

de données facilitant la mise en pair à pair. Toutefois, notre étude ne s'attardera pas sur cet aspect. Nous abstrayons la structure de données derrière une interface pour nos définitions, et nos simulations s'appuient sur une liste doublement liée (DLL) [1]. Nous conservons cette même structure pour toutes les simulations afin de focaliser nos analyses sur les aspects de mise à l'échelle.

En outre, les théorèmes développés doivent être suffisamment généralistes pour fonctionner indépendamment du type d'utilisateur. Cela signifie que la complexité d'une transaction sur Morpho doit rester constante, quelle que soit la situation. Cette constance est essentielle car tout blocage imprévu dû à un cas de figure spécifique pourrait compromettre la totalité du protocole sur une plateforme aussi immuable qu'Ethereum.

Enfin, notre conception de la mise à l'échelle de Morpho doit prendre en compte quatre paramètres essentiels détaillés dans le livre blanc, qui guideront la conception du protocole pour garantir ses promesses (voir Tableau 1.1 pour un résumé).

Tableau 1.1 Spécifications de Morpho à garantir

Paramètre	**Description**	**Spécification technique**
Efficacité économique	Garantir l'efficacité maximale du protocole en termes de taux d'intérêt pour l'utilisateur.	Maximiser le pourcentage d'utilisateurs en pair à pair.
Efficacité du coût d'une transaction	Réduire au maximum le nombre d'opérations pour garantir une efficacité dans l'exécution du protocole.	Mener une étude en terme de coût en gas du protocole.
Simplicité	Le fonctionnement en boîte noire du protocole doit être similaire à l'état de l'art des protocoles de prêt et d'emprunts.	La mise à l'échelle ne doit pas exiger une connaissance supplémentaire de l'utilisateur.
Équité	L'optimisation pair à pair de Morpho doit être accessible au plus grand nombre d'utilisateurs possible et s'adapter à la diversité, notamment en termes de montant déposé/emprunté.	La mise à l'échelle ne doit pas favoriser un acteur plus qu'un autre.

1. https ://en.wikipedia.org/wiki/Doubly_linked_list

1.6 Plan du mémoire

Le présent mémoire est structuré comme suit :

— **Chapitre 2 : Revue de Littérature** - Ce chapitre propose un panorama des sujets essentiels à la compréhension de ce mémoire, notamment : (1) l'émergence et les spécificités des blockchains publiques, avec une attention particulière portée à Ethereum ; (2) les implications de ces technologies dans le domaine financier et l'évolution vers la finance décentralisée ; (3) les protocoles de prêt et d'emprunt au sein de la finance décentralisée, leur fonctionnement et leurs limites.

— **Chapitre 3 : Étude Formelle de Morpho** - Une analyse détaillée du protocole Morpho est présentée ici, incluant des définitions, des propriétés, des théorèmes et une simulation mettant en évidence les difficultés de mise à l'échelle de Morpho.

— **Chapitre 4 : Mécanisme Delta** - Ce chapitre décrit le mécanisme Delta, une solution proposée pour surmonter les défis identifiés dans le chapitre précédent. Des exemples concrets et des pseudocodes sont fournis pour illustrer son fonctionnement et faciliter son implémentation.

— **Chapitre 5 : Étude Comparative et Validation** - Une comparaison de Morpho avec d'autres protocoles de prêt et d'emprunt actuels, tels qu'AaveV2, est effectuée dans ce chapitre. L'accent est mis sur l'évaluation de l'efficacité de Morpho par rapport aux solutions existantes.

— **Chapitre 6 : Conclusion et Perspectives** - Les principales conclusions de cette recherche sont résumées ici. Les applications concrètes de ce travail et les orientations futures pour la recherche et le développement de Morpho sont également discutées.

CHAPITRE 2 CONCEPTS DE BASES ET REVUE DE LITTÉRATURE

Le monde du prêt et de l'emprunt de devises numériques a connu une évolution assez impressionnante, influençant à la fois l'univers des cryptomonnaies et celui de la finance traditionnelle. Les avancées technologiques dans le domaine des chaînes de blocs publiques ont permis de rendre les taux plus compétitifs grâce à la transparence des calculs sur les protocoles blockchain [7]. Cela a également donné une nouvelle utilité aux devises numériques, qui peuvent désormais être utilisées pour des opérations financières. Ces développements ont été rendus possibles grâce à l'évolution d'Ethereum [1].

2.1 De Bitcoin à Ethereum

Le 3 janvier 2009 à 19 :15 :05 GMT+1, le premier bloc de Bitcoin [8] a été miné, suscitant la curiosité de nombreux passionnés. L'idée initiale de Bitcoin est simple : transférer de la valeur au sein d'un système distribué et immuable, dans le but de redistribuer la valeur aux utilisateurs à travers une base de données publique, non permise et décentralisée. Bitcoin utilise la cryptographie, un réseau pair-à-pair et une blockchain pour assurer la véracité des données enregistrées sur la chaîne de blocs.

Cependant, les applications étaient alors limitées et se cantonnaient au transfert de bitcoins d'un utilisateur à un autre. L'objectif ultime et la motivation derrière le développement de Bitcoin étaient d'assurer la résilience et la valeur des données sauvegardées sur la blockchain, en contraste avec les serveurs bancaires, qui sont peu transparents malgré les nombreuses réglementations et audits auxquels ils sont soumis [7].

Au fil des premières années, Bitcoin a gagné en popularité et en utilisation. Toutefois, la communauté n'était pas prête à évoluer. Éviter les changements, garantir la sécurité et maintenir un mécanisme résilient étaient des arguments avancés par la communauté, freinant les développements de Bitcoin [9].

Quelques années plus tard, en 2016, Vitalik Buterin propose dans le livre blanc d'Ethereum [1] une évolution prometteuse de Bitcoin. Ethereum partage les principes de Bitcoin mais ajoute une fonctionnalité majeure : les transactions qui constituent un bloc peuvent maintenant exécuter une logique spécifique avant d'être validées, et le résultat de cette exécution est stocké dans un espace partagé (voir Figure 2.1). Cette logique s'exécute dans un environnement dit Turing-complet [10], ce qui augmente considérablement les possibilités offertes par la chaîne de blocs publique.

Vitalik propose aussi un plan d'évolution de la blockchain, passant d'un modèle simpliste à un modèle plus robuste en termes de mise à l'échelle, de coûts pour l'utilisateur et l'environnement [11], ainsi que de simplicité d'implémentation. Pour initier le projet, Ethereum a commencé avec une première version simple, similaire à Bitcoin sur le plan cryptographique, mais se concentrant principalement sur la machine d'exécution embarquée, l'EVM.

Ethereum s'appuie sur le même modèle communautaire que Bitcoin pour faire évoluer le protocole, et les modifications ou "forks" sont proposés et acceptés (ou refusés) par le biais des EIP (Ethereum Improvement Proposals) [12].

2.1.1 La machine virtuelle d'Ethereum

Ethereum peut être vu comme une machine à états, maintenue par des ordinateurs du monde entier. Les transactions, regroupées en blocs, représentent des transitions valides entre les états [13]. Cet état est représenté par une base de données clé-valeur distribuée, où la clé est l'adresse d'un compte et la valeur est un objet contenant le solde en ether (la cryptomonnaie d'Ethereum) ainsi que le code et le stockage associés au contrat intelligent de ce compte.

Les transactions effectuées entre les comptes changent l'état de la machine virtuelle d'Ethereum (EVM). La figure 2.1 montre comment les transactions sont traitées par l'EVM. Chaque transaction peut comporter de l'Ether, des données (éventuellement du code d'un contrat intelligent) et doit être signée par la clé privée de l'expéditeur pour être valide.

Nous allons simplement noter Σ l'ensemble des états valides de la blockchain. Par exemple, prenons $\sigma_i \in \Sigma$ un état valide du stockage d'Ethereum a un block donné i, alors la transition vers $\sigma_{i+1} \in \Sigma$ peut se faire uniquement avec la reduction de transactions valides.

De plus, pour un bloc donné i, tous les noeuds Ethereum partagent le même état σ_i. D'ou l'aspect déterministe d'Ethereum.

L'EVM est une partie essentielle de l'écosystème Ethereum. Il s'agit d'un environnement d'exécution isolé, s'exécutant sur tous les nœuds du réseau, qui traite le code des contrats intelligents. L'environnement est Turing-complet, ce qui signifie qu'il peut exécuter tout algorithme, étant donné assez de temps et de ressources. Cette caractéristique de l'EVM rend Ethereum plus puissant et polyvalent que Bitcoin.

2.2 Les Smart Contracts

Avec l'introduction des contrats intelligents, Ethereum a ouvert la voie à une nouvelle génération d'applications décentralisées. Un contrat intelligent est un programme qui s'exécute

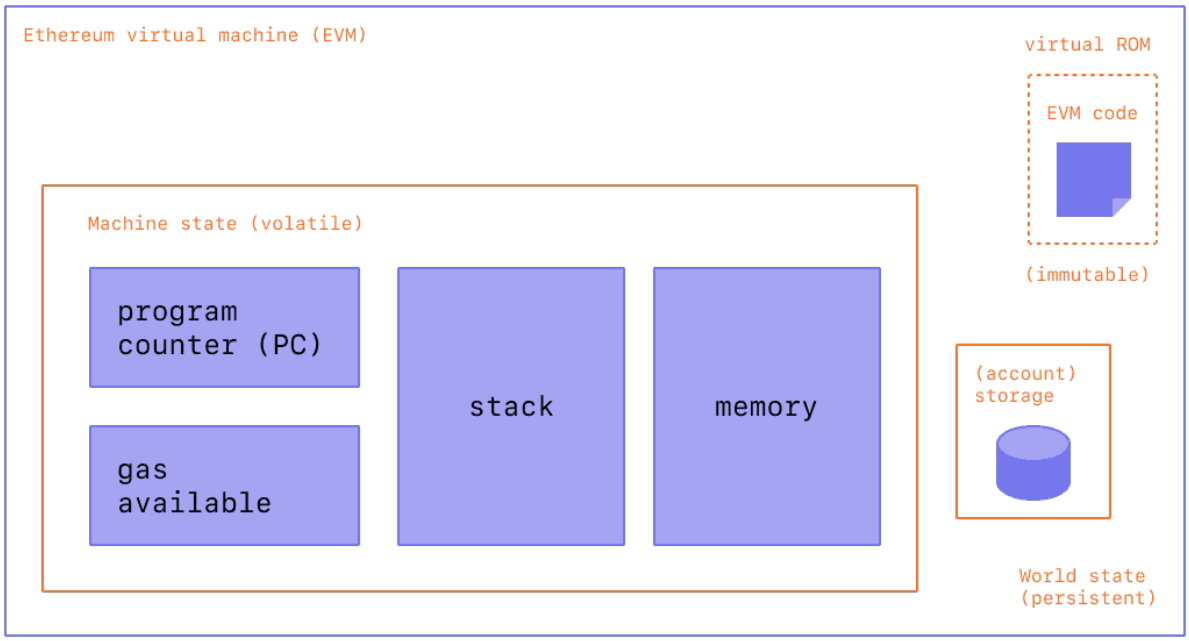


FIGURE 2.1 Le traitement des transactions par la Machine Virtuelle d'Ethereum.

sur la blockchain Ethereum. Il est déployé sous forme de bytecode dans la blockchain et peut être interagi avec, en envoyant des transactions à l'adresse du contrat [14].

Les contrats intelligents permettent de créer des applications qui fonctionnent exactement comme programmées, sans possibilité de temps d'arrêt, de censure, de fraude ou d'interférence de tiers [1]. Ces applications peuvent aller de la simple gestion d'un jeton numérique à la réalisation d'opérations financières complexes [15].

Les contrats intelligents ont évolué pour devenir des composants centraux de nombreux systèmes décentralisés, tels que les applications financières décentralisées (DeFi), les jeux, les marchés d'actifs numériques et bien plus encore.

2.3 Les Protocoles de prêts et d'emprunts sur Ethereum

Dans un univers décentralisé comme celui de la DeFi, tous les protocoles se doivent d'être automatisés et sans intermédiaire. Il devient donc compliqué de savoir comment contracter un prêt auprès d'un contrat intelligent, là où en finance traditionnelle, un intermédiaire (généralement une banque) va passer du temps à étudier un profil, faire une étude de risque et prêter (ou non) un certain montant à un taux guidé par les marchés financiers et les régulateurs.

Les premières implémentations de protocoles se sont fixées sur un modèle sans intermédiaire.

Définition 2.3.0.1. *Les protocoles de prêt et d'emprunts (formellement appelés "Protocol for Loanable Funds" ou PLF) en finance décentralisée que nous étudierons sont dits* ***sur-collatéralisés****. Cela signifie qu'un emprunteur doit en premier lieu déposer un montant relativement plus important pour pouvoir contracter un prêt [16].*

Prenons un exemple concret pour illustrer cela :

Exemple 2.3.0.1. *Alice souhaite emprunter 80 000 DAI sur le protocole P1. Le protocole accepte de lui prêter ce montant uniquement si elle offre un dépôt minimum d'un équivalent de 100 000$. Alice dépose donc une représentation numérique de son bien immobilier, qui a une valorisation de marché de 110 000$. Ainsi, Alice peut emprunter les 80 000 DAI et aller faire des travaux de rénovation dans sa maison.*

Le lien entre le montant empruntable et le dépôt initial souhaité est appelé taux de collatéralisation. Dans l'exemple précédent, il est de 80% et définit le risque de variation du prix accepté par les prêteurs. Ce taux est associé à un collatéral et est assez difficile à définir pour un protocole autonome.

Expliquons tout d'abord pourquoi ce taux définit le risque des prêteurs. Pour cela, nous devons d'abord définir ce qu'il se passe si le prix du collatéral descend de telle manière à placer l'emprunteur en dessous de ce facteur. Pour quantifier cela, on introduit une mesure de facteur de santé d'une position, ou "Health Factor" en anglais [4] :

Définition 2.3.0.2. *Le facteur de santé (ou "Health Factor" en anglais) représente le taux de collatéralisation d'un utilisateur à un instant t.*

Notons s_u la valeur du dépôt (la supply) de l'utilisateur, λ son taux de collatéralisation, et b_u sa dette (borrow en anglais) :

$$hf(t) = \frac{s_u(t) \times \lambda}{b_u(t)} \tag{2.1}$$

s_u et b_u sont variables dans le temps et évoluent en fonction du prix des devises utilisées, ainsi que des taux d'intérêts.

Ainsi, si le HF descend en dessous de 1, alors un utilisateur devient liquidable.

Exemple 2.3.0.2. *Reprenons le cas d'Alice défini précédemment. Son facteur de santé peut être calculé ainsi :*

Utilisateur	*Total déposé*	*Total collatéral*	*Total Emprunté*	*HF*
Alice	*110 000$*	*88 000$*	*80 000$*	*1.1*

Le HF est supérieur à 1, donc la position d'Alice est actuellement sûre. Si la valeur de son bien immobilier devait descendre à 90 000$, alors son HF deviendrait :

$$hf(t) = \frac{90000\$ \times 0.8}{80000\$} = 0.9 \tag{2.2}$$

À ce moment, Alice deviendrait liquidable.

La liquidation est un élément clé du risque de crédit en DeFi. Elle est le mécanisme par lequel un emprunteur peut se voir retirer une partie ou la totalité de son collatéral en cas de baisse du facteur de santé en dessous de 1. La liquidation peut être déclenchée par n'importe quel participant au réseau qui observe que le HF d'un utilisateur est inférieur au seuil autorisé. Le liquidateur est alors récompensé par une fraction du collatéral liquidé comme incitation à effectuer cette opération nécessaire au bon fonctionnement du protocole [17].

Maintenant, si Bob ou un autre liquidateur ne liquide pas la position assez vite et que la position de Alice devient sous-collatéralisée (les collatéraux ne couvrent plus les dettes), la position devient insolvable.

La liquidation est en réalité une fenêtre d'arbitrage très compétitive. De plus, lorsque la liquidation d'un utilisateur est proche, il existe de nombreux moyens pour essayer de rendre cette position liquidable, en ordonnant des transactions, en manipulant les marchés ou encore par bien d'autres méthodes. Tout cet aspect compétitif est analysé en profondeur dans le papier cité plus tôt de Simtopia [18].

Reprenons l'exemple d'Alice :

Exemple 2.3.0.3. *L'argent est fourni par des prêteurs qui acceptent de s'exposer à une variation de prix de sa maison de 20%. Alice sera en situation de liquidation si sa maison atteint un prix de marché de 100000$, et le bien sera redistribué aux prêteurs.*

Comme nous l'avons vu, il y a un facteur d'exposition de 10% accepté par Alice, et un facteur de variation de 20% accepté par les prêteurs. Les 10% sont choisis par Alice au moment où elle a décidé du montant qu'elle empruntait.

Le taux de collatéralisation de 80% est choisi par le protocole lui-même, en fonction d'analyses de risque face à l'état des marchés. Ces valeurs peuvent généralement être définies et votées par la gouvernance.

Le facteur de liquidation

Le facteur de liquidation (noté lf) représente le pourcentage que le liquidateur va recevoir en récompense pour liquider une position. Plus ce facteur est élevé, plus la liquidation sera profitable pour le liquidateur, mais cela peut également accroître la mauvaise dette (voir section 2.3).

Sur les protocoles existants, il existe deux types de facteurs : les facteurs dynamiques et les facteurs statiques.

Les facteurs statiques sont définis pour chaque marché et sont les plus simples à comprendre.

Exemple 2.3.0.4. *Alice a un emprunt de 900$ en DAI, collatéralisé par 1000$ en ETHER. Le taux de collatéralisation de l'ETHER est de 85%. Alice a donc un facteur de santé de 0.94 et elle est donc liquidable.*

Le facteur de liquidation du marché ETHER vaut 5%. Ainsi, la liquidation de la position d'Alice permettra de rembourser 900$ de DAI et de récupérer $900 \times 1.05 = 945\$$ *en ETHER.*

La position résultante d'Alice est un dépôt de 55$ en ETHER, et plus aucune dette.

C'est ce modèle qui est choisi par Aave.

Les facteurs dynamiques, quant à eux, évoluent avec le facteur de santé : plus la position se dégrade, plus la récompense est grande. Ainsi, la position est liquidée pour un minimum de profit pour le liquidateur, ce modèle est économiquement beaucoup plus avantageux pour les utilisateurs [19].

Les positions insolvables

La liquidation est une question de rapidité. Si une position n'est pas liquidée assez vite (due à des changements de prix très rapides entre le collatéral et la dette, par exemple), alors on dit que la position devient insolvable (voir Figure 2.2). Ce type de position se résume à un utilisateur qui possède uniquement de la dette.

Exemple 2.3.0.5. *Alice a un emprunt de 1000$ en DAI, collatéralisé par 1000$ en ETHER. Le taux de collatéralisation de l'ETHER est de 85%. Alice a donc un facteur de santé de 0.85 et elle est donc liquidable.*

Supposons que le facteur de liquidation est de 5%, alors si je calcule la dette à rembourser pour récupérer tout le collatéral en tant que liquidateur, j'obtiens 952$. Ainsi, liquider cette position va résulter en une dette non réalisée pour Alice de 48$.

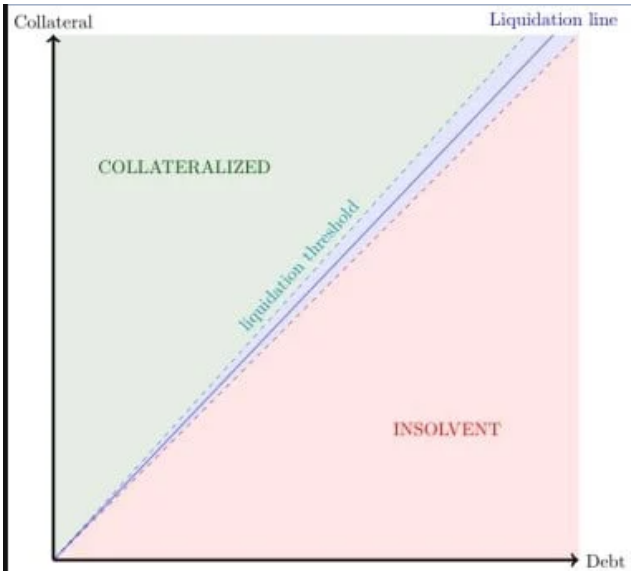


FIGURE 2.2 Collatéralisation et liquidation

Propriété 2.3.0.1. *Soit u un utilisateur à un instant t, c_u son collatéral en \$, et b_u sa dette en \$. Notons également lf le facteur de liquidation du marché en question. La liquidation de cette position va créer de la mauvaise dette si et seulement si*

$$b_u > \frac{c_u}{1 + lf} \tag{2.3}$$

De nombreuses études ont été faites sur la mauvaise dette, notamment au sein de Morpho Labs. En particulier, Aave utilise une réserve de liquidité qui est provisionnée par un frais sur l'utilisation du protocole. En cas de mauvaise dette, cette réserve peut être utilisée afin de repayer la mauvaise dette et d'éviter le phénomène de *bank run*. En effet, cette dette est mutualisée dans ce qu'on appelle une "pool". La liquidité empruntée est provisionnée par d'autres utilisateurs, et c'est le dernier à quitter le marché qui subit la mauvaise dette et qui ne peut pas récupérer ses fonds [20].

2.3.1 Pool de liquidité

Les fonds empruntés peuvent être fournis par divers participants, chacun finançant une part de l'emprunt. Ce système est qualifié de prêt mutualisé et est défini comme un modèle en « pool » : les prêteurs alimentent une « pool » de liquidité, dont se servent les emprunteurs. Ce modèle, introduit en 2016 par Compound, représente actuellement la norme pour les protocoles de prêt et d'emprunt. Dans cette section, nous décrirons son fonctionnement et expliquerons pourquoi il est particulièrement efficace sur Ethereum. [4, 21]

Description

Nous allons décrire comment fonctionne les pool de liquidités sur Ethereum. Ceci nous servira de base pour comprendre la construction de Morpho et les contraintes de mise à l'échelle.

Dans un premier temps, nous allons définir une notation qui sera utilisée tout au long du mémoire.

Soit s_a^u le dépôt (supply) d'un utilisateur u dans la pool a, et b_a^u sa dette. Désignons par U_a l'ensemble des utilisateurs de la pool a, S_a la somme totale des dépôts et B_a la dette totale, définis comme suit :

$$\begin{aligned} S_a &= \textstyle\sum_{u \in U_a} s_a^u, \\ B_a &= \textstyle\sum_{u \in U_a} b_a^u. \end{aligned} \tag{2.4}$$

Considérons l'exemple suivant du marché DAI, avec un taux de collatéralisation de 82% :

Utilisateur	Total déposé	Total collatéral	Total emprunté	HF
Alice	110,000 DAI	$110,000 \times 82\% = 90,200\$$	53 ETH $= 80,000\$$	1.1
Bob	40,000 DAI	$40,000 \times 82\% = 32,800\$$	0	Infini
Charlie	60,000 DAI	$60,000 \times 82\% = 49,200\$$	2 ETH $= 3,000\$$	16.4
Carole	100 ETH	-	0	Infini

Dans cet exemple, $S_{DAI} = 210,000$ DAI et $B_{ETH} = 55$ ETH.

DAI et ETH représentent deux pools distinctes. Ainsi, la liquidité empruntée dans la pool ETH provient d'autres prêteurs.

Taux d'utilisation

Définition 2.3.1.1. *Le taux d'utilisation d'une pool a (U_a) est le rapport du capital emprunté sur le capital déposé. Notons S_a la somme des dépôts de tous les utilisateurs et B_a la somme des emprunts, ce qui donne :*

$$U_a = \frac{B_a}{S_a} \in [0, 1]. \tag{2.5}$$

Un enjeu essentiel pour un protocole de liquidité est la disponibilité du capital, c'est-à-dire la capacité pour les utilisateurs d'emprunter ou de retirer leurs fonds à tout moment. Le capital disponible est alors :

$$S_a - B_a = (1 - U_a) \cdot S_a. \tag{2.6}$$

Dans notre exemple, le taux d'utilisation de la pool ETH est de 55%, avec un dépôt total de 100 ETH, ce qui donne un capital disponible de 45 ETH. Les utilisateurs de cette pool (ici exclusivement Carole) peuvent retirer jusqu'à 45 ETH.

Il est donc nécessaire d'inciter à la fois Alice et Charlie à rembourser leur prêt et d'attirer de nouveaux participants. Inversement, lorsque l'utilisation est faible, il est souhaitable d'encourager les emprunts sans forcément attirer de nouveaux fournisseurs de liquidité. Pour ce faire, les protocoles modulent les taux d'intérêt.

Taux d'intérêts

Les protocoles se distinguent principalement par leur modèle d'intérêts, et certains sont reconnus comme étant plus robustes que d'autres (voir l'étude liée). Examinons le fonctionnement du protocole Aave.

Définition 2.3.1.2. *Le taux d'emprunt linéaire r^B_{linear} sur Aave est défini comme suit*[1] *:*

$$r^B_{linear} = \begin{cases} R_0 + \frac{U}{U_{opt}} R_1 & si\ U \leq U_{opt}, \\ R_0 + R_1 + \frac{U - U_{opt}}{1 - U_{opt}} R_2 & si\ U > U_{opt}, \end{cases} \tag{2.7}$$

où R_0, R_1 et R_2 sont des constantes réelles positives déterminées par la gouvernance du protocole, R_2 étant nettement supérieur à R_0 et R_1. Les taux augmentent donc progressivement avec le taux d'utilisation jusqu'au seuil U_{opt}, puis s'accroissent très rapidement au-delà, pouvant atteindre des taux supérieurs à 50% par an.

Les intérêts sont composés à chaque seconde, donc le taux annualisé est calculé selon la formule suivante :

$$r^B = \left(1 + \frac{r^B_{linear}}{n_y}\right)^{n_y} - 1, \tag{2.8}$$

avec n_y représentant le nombre de secondes par an.

Les taux d'intérêts sont fondés sur le taux d'utilisation et un taux d'utilisation optimal, noté U_{opt}.

Voyons à présent comment les intérêts versés par les emprunteurs sont redistribués aux prêteurs.

Les prêteurs perçoivent des revenus équivalents à ceux payés par l'ensemble des emprunteurs,

1. Voir : https ://github.com/aave/protocol-v2/blob/master/contracts/protocol/lendingpool/DefaultReserveInterestRateS

ce qui fait que les taux de prêt sont directement liés aux taux d'emprunt. Cette corrélation nous fournit la relation suivante entre les deux taux [2] :

$$r^S_{linear} = U \cdot r^B_{linear}, \tag{2.9}$$

d'où :

$$r^S_{linear} = \begin{cases} U \cdot \left(R_0 + \frac{U}{U_{opt}} R_1\right) & \text{si } U \leq U_{opt}, \\ U \cdot \left(R_0 + R_1 + \frac{U - U_{opt}}{1 - U_{opt}} R_2\right) & \text{si } U > U_{opt}. \end{cases} \tag{2.10}$$

Les taux annualisés sont calculés avec la même formule que celle utilisée pour les taux d'emprunt.

Exemple 2.3.1.1. *Prenons l'exemple du DAI. Les paramètres pour ce marché sur Aave sont les suivants :*

	U_{opt}	R_0	R_1	R_2
DAI	*80%*	*0%*	*4%*	*75%*

Les figures 2.3 et 2.4 illustrent les taux d'intérêts en fonction du taux d'utilisation U pour le DAI.

À retenir : les taux d'intérêts sont déterminés en fonction du taux d'utilisation. Il s'agit d'une décision propre à chaque protocole que nous n'analyserons pas ici, car cela relève de l'analyse des risques financiers, ce qui est hors de notre champ d'intérêt en génie logiciel.

Pour accumuler les taux d'intérêts, nous introduirons la notion d'indice, qui est également utilisée en finance traditionnelle.

Les Indices

Un indice est une valeur strictement croissante qui représente la distribution des intérêts au sein d'une pool. En d'autres termes, cela permet de distribuer de manière intégrale (taux d'intérêts $\times$ temps) les intérêts à tous les utilisateurs.

Examinons comment cela fonctionne en pratique :

Soit $(r_i)_{i \in \mathbb{N}}$ une suite de taux et $(\lambda_i)_{i \in \mathbb{N}}$ la suite correspondante d'indices.

Soit $(t_i)_{i \in \mathbb{N}}$ une séquence croissante de moments où l'indice est mis à jour. Nous posons que $t_0 = 0$.

2. Il est à noter qu'en pratique, le protocole prélève des frais sur ces taux d'intérêts.

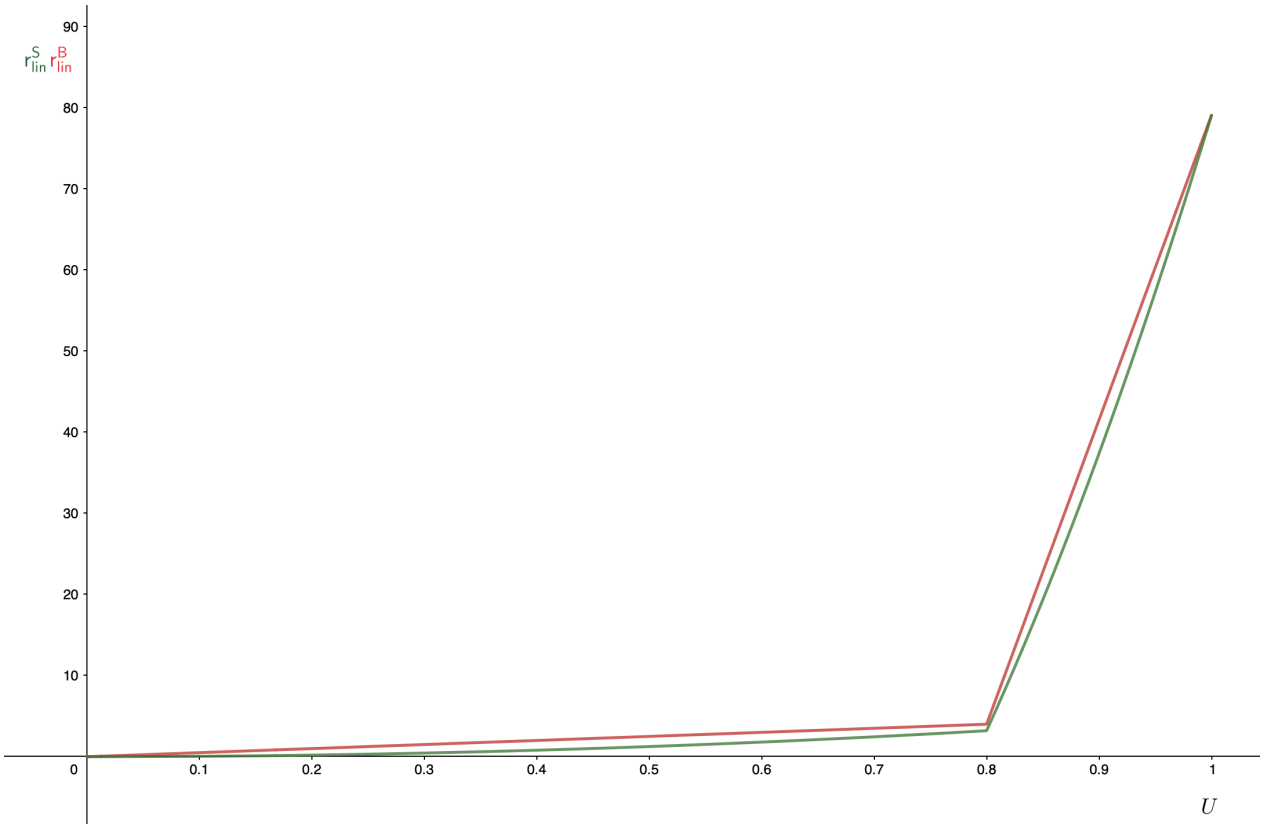


FIGURE 2.3 Taux d'intérêts du DAI sur Aave.

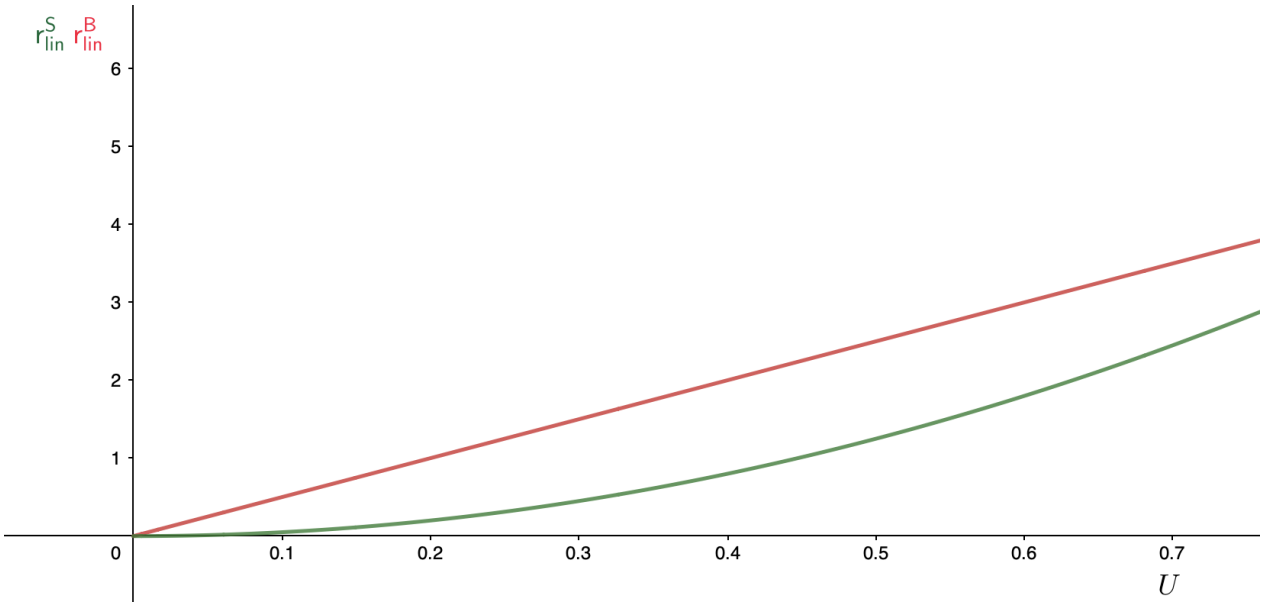


FIGURE 2.4 Taux d'intérêts du DAI sur Aave, pour $U \in [0, 0.7]$.

Soit t_{courant} un moment de mise à jour avec courant > 0, on écrit $t_{\text{dernier}} = t_{\text{courant}-1}$ et $\Delta t = t_{\text{courant}} - t_{\text{dernier}}$. Nous décrivons l'évolution de l'indice selon les trois paramètres suivants, la suite (λ_i) étant soit (λ_i^{lin}), $(\lambda_i^{\text{comp}})$, ou $(\lambda_i^{\text{recomp}})$.

Avec des intérêts linéaires. La formule d'évolution de l'indice en une étape est donnée par :

$$\forall t \in \mathbb{N}, \quad \lambda_{t+1}^{\text{lin}} = \lambda_t^{\text{lin}} + \lambda_0^{\text{lin}} \cdot r_t$$

Par récurrence, on peut montrer que :

$$\forall t \in \mathbb{N}, \quad \lambda_t^{\text{lin}} = \lambda_0^{\text{lin}}(1 + \sum_{k<t} r_k)$$

En supposant que (r_i) est constant sur $[t_{\text{dernier}}, t_{\text{courant}}[$, la formule suivante permet de mettre à jour l'indice dans le cas d'intérêts linéaires :

$$\lambda_{t_{\text{courant}}}^{\text{lin}} = \lambda_{t_{\text{dernier}}}^{\text{lin}} + \Delta t \cdot \lambda_0^{\text{lin}} \cdot r_{t_{\text{dernier}}}$$

Avec intérêts composés. La formule d'évolution de l'indice en une étape est donnée par :

$$\forall t \in \mathbb{N}, \quad \lambda_{t+1}^{\text{comp}} = \lambda_t^{\text{comp}} \cdot (1 + r_t)$$

Par récurrence, on peut montrer que :

$$\forall t \in \mathbb{N}, \forall t' \in \mathbb{N}, t' < t, \quad \lambda_t^{\text{comp}} = \lambda_{t'}^{\text{comp}} \prod_{t' \leq i < t} (1 + r_i)$$

En supposant que (r_i) est constant sur $[t_{\text{dernier}}, t_{\text{courant}}[$, la formule suivante permet de mettre à jour l'indice dans le cas d'intérêts composés :

$$\lambda_{t_{\text{courant}}}^{\text{comp}} = \lambda_{t_{\text{dernier}}}^{\text{comp}} (1 + r_{t_{\text{dernier}}})^{\Delta t}$$

Avec des intérêts linéaires recomposés. La formule d'évolution de l'indice en une étape est donnée par :

$$\forall t \in \mathbb{N}, \quad \lambda_{t+1}^{\text{recomp}} = \lambda_t^{\text{recomp}} + \lambda_{t'}^{\text{recomp}} \cdot r_t$$

où t' est le dernier moment de mise à jour avant t.

Par récurrence, on peut prouver que, pour $s, e \in \mathbb{N}$ tels que $s \leq e$:

$$\lambda_e^{\text{recomp}} = \lambda_s^{\text{recomp}} \prod_{s \leq i < e} \left(1 + \sum_{t_i \leq j < t_{i+1}} r_j\right)$$

Et en particulier :

$$\lambda_{t_{\text{courant}}}^{\text{recomp}} = \lambda_{t_{\text{dernier}}}^{\text{recomp}} \left(1 + \sum_{t_{\text{dernier}} \leq j < t_{\text{courant}}} r_j\right)$$

En supposant que (r_j) est constant sur $[t_{\text{dernier}}, t_{\text{courant}}[$, la formule suivante permet de mettre à jour l'indice dans le cas d'intérêts recomposés :

$$\lambda_{t_{\text{courant}}}^{\text{recomp}} = \lambda_{t_{\text{dernier}}}^{\text{recomp}} \left(1 + \Delta t \cdot r_{t_{\text{dernier}}}\right)$$

Bénéfices Le principal avantage de cette méthode est de pouvoir distribuer les intérêts à tous les utilisateurs en mettant à jour une seule valeur, plutôt que de mettre à jour chaque utilisateur individuellement. C'est ce mécanisme qui assure la mise à l'échelle du modèle orienté pool de liquidité. En effet, les intérêts accumulés ne sont pas dépendants de la fréquence de mise à jour, ni du nombre et de la variété d'utilisateurs.

Notation Indépendamment de l'implémentation des taux choisie, nous utiliserons la notation suivante pour déterminer les indices de prêt et d'emprunt d'une pool donnée :

— λ_θ^S : Indice sur la pool pour les prêteurs (Suppliers)
— λ_θ^B : Indice sur la pool pour les emprunteurs (Borrowers)

L'implémentation de ce modèle permet de récupérer la balance d'un utilisateur avec ses intérêts accumulés à n'importe quel moment en procédant ainsi :

Récupération de la balance d'un utilisateur Pour récupérer la balance actualisée d'un utilisateur, on utilise les indices de prêt ou d'emprunt, selon que l'utilisateur est un prêteur ou un emprunteur. Le processus est le suivant :

— Au moment où l'utilisateur dépose dans la pool a un montant m à $t = t_0$, nous enregistrons sa balance $b_u^S(a, t_0) = \frac{m}{\lambda_\theta^S(t_0)}$.
— À tout moment $t_1 \geq t_0$, la balance de l'utilisateur comprenant les intérêts vaut $b_u^S(a, t_1) = b_u^S(a, t_0) \times \lambda_\theta^S(t_1)$.
— Les intérêts accumulés par l'utilisateur entre t_0 et t_1 valent $i = b_u^S(a, t_1) - b_u^S(a, t_0) = b_u^S(a, t_0) \times (\lambda_\theta^S(t_1) - 1) = \frac{m \times (\lambda_\theta^S(t_1) - 1)}{\lambda_\theta^S(t_0)}$.

On peut observer que la balance de l'utilisateur peut être calculée à n'importe quel instant t, en ayant uniquement modifié l'indice du marché. C'est ainsi que le modèle de pool peut aller à l'échelle sur Ethereum. En effet, chaque interaction avec la pool va mettre à jour l'indice du marché, en fonction d'un taux d'intérêt, et tous les utilisateurs ont instantanément accumulé des intérêts.

2.3.2 Fonctionnalités Clés des Protocoles de Prêt Décentralisés

Les protocoles de prêt décentralisés comme Aave et Compound offrent des mécanismes financiers automatisés et transparents sans intermédiaires. Ces plateformes reposent sur des contrats intelligents pour fournir des services financiers tels que le dépôt de garantie, l'emprunt de fonds, le retrait, le remboursement de prêts et la liquidation des garanties en cas de défaut de paiement. Dans cette section, nous allons examiner de plus près comment les cinq fonctions classiques de ces protocoles interagissent pour créer un système financier décentralisé et efficace.

Supply (Fourniture de liquidités)

La fonction *Supply* permet aux utilisateurs de déposer leurs actifs cryptographiques dans un pool de liquidité pour gagner des intérêts. Ces actifs déposés peuvent servir ensuite de garantie pour des prêts et permettent aux fournisseurs de liquidités de recevoir un rendement.

Pseudo-code

```
fonction supply(utilisateur, montant, actif):
    pool_de_liquidité[actif] += montant
    solde_de_l'utilisateur[actif] -= montant
    émettre cToken/utilisateurToken à l'utilisateur
```

Exemple Alice dépose 100 DAI dans le pool de liquidité et reçoit des cDAI en retour, qui représentent sa part dans le pool.

Borrow (Emprunt)

Borrow est la fonctionnalité qui permet aux utilisateurs d'emprunter des actifs à partir des pools de liquidité. Les emprunteurs doivent fournir une garantie suffisante pour sécuriser le prêt et sont tenus de payer des intérêts sur le montant emprunté.

Pseudo-code

```
fonction borrow(utilisateur, montant, actif):
    si solde_de_l'utilisateur[actif] >= montant * facteur_de_collatéral:
        solde_de_l'utilisateur[actif] -= montant
        dette_de_l'utilisateur[actif] += montant
        transférer actif à l'utilisateur
    sinon:
        échec de la transaction
```

Exemple Bob souhaite emprunter 50 DAI et doit fournir une garantie suffisante selon le facteur de collatéral exigé.

Withdraw (Retrait)

La fonction *Withdraw* est utilisée pour retirer des actifs précédemment fournis à la pool de liquidité. Les utilisateurs peuvent retirer jusqu'à la valeur de leur garantie moins le montant du prêt en cours, s'il y en a un.

Pseudo-code

```
fonction withdraw(utilisateur, montant, actif):
    si montant <= solde_de_l'utilisateur[actif] - dette_de_l'utilisateur[actif]:
        pool_de_liquidité[actif] -= montant
        solde_de_l'utilisateur[actif] += montant
        brûler cToken/utilisateurToken de l'utilisateur
    sinon:
        échec de la transaction
```

Exemple Alice décide de retirer 50 DAI du pool. Elle doit avoir suffisamment de cDAI et maintenir la garantie pour ses dettes en cours.

Repay (Remboursement)

Le remboursement d'un prêt est réalisé par la fonction *Repay*. Les emprunteurs remboursent le capital emprunté ainsi que les intérêts accumulés pour libérer ou récupérer leur garantie.

Pseudo-code

```
fonction repay(utilisateur, montant, actif):
    solde_de_l'utilisateur[actif] -= montant
    dette_de_l'utilisateur[actif] -= montant
    transférer actif du utilisateur au pool de liquidité
```

Exemple Bob rembourse 25 DAI de sa dette pour récupérer une partie de sa garantie ou améliorer son ratio de collatéral.

Liquidate (Liquidation)

Si un emprunteur ne maintient pas un ratio de garantie adéquat et que la valeur de sa garantie chute en dessous d'un certain seuil, la fonction *Liquidate* permet à d'autres utilisateurs, souvent appelés liquidateurs, de rembourser une partie ou la totalité du prêt en défaut en échange d'une partie de la garantie à un prix réduit.

Pseudo-code

```
fonction liquidate(liquidateur, utilisateur, dette, actif):
    si ratio_de_collatéral_de_l'utilisateur < seuil_minimum:
        montant_à_liquider = calculer_montant_à_liquider(dette)
        solde_de_l'utilisateur[actif] -= montant_à_liquider
        transférer actif du liquidateur au pool de liquidité
        transférer garantie à liquidateur
    sinon:
        échec de la transaction
```

Exemple Charlie utilise la fonction de liquidation pour rembourser 30 DAI de la dette de Bob et reçoit en échange une partie de la garantie de Bob à un prix avantageux.

2.3.3 Le coût des transactions

La question des coûts de transaction sur les blockchains comme Ethereum est centrale, notamment lorsqu'il s'agit d'opérations complexes comme les emprunts et les retraits dans le cadre de protocoles de finance décentralisée (DeFi). Lorsqu'un utilisateur effectue un emprunt ou un retrait, le protocole doit effectuer plusieurs vérifications pour s'assurer que l'opération ne met pas en danger la santé financière de la plateforme ni celle de l'utilisateur.

Pour les emprunts, le protocole doit vérifier que l'utilisateur a suffisamment de collatéral pour couvrir le nouvel emprunt. Cela inclut non seulement le montant du nouvel emprunt, mais aussi tous les autres emprunts que l'utilisateur pourrait avoir sur d'autres marchés. Plus un utilisateur a de positions diversifiées sur différents marchés, plus le nombre de vérifications nécessaires est élevé, ce qui augmente le coût du gas.

Pour les retraits, le protocole doit s'assurer que le retrait d'un certain montant de collatéral ne va pas rendre l'utilisateur insolvable. Encore une fois, cela nécessite de vérifier toutes les obligations de l'utilisateur à travers tous les marchés sur lesquels il est actif. Chaque marché peut avoir son propre facteur de collatéralisation et ses propres conditions de santé financière, ce qui rend les calculs encore plus complexes.

La vérification de la santé financière d'une position (souvent appelée "health check") est une opération qui demande beaucoup de calculs et donc consomme beaucoup de gas. Le gas est une mesure de la quantité de ressources de calcul que l'opération nécessite sur la blockchain. Plus les calculs sont complexes et nombreux, plus le coût en gas sera élevé.

En bref, lorsqu'un utilisateur a des positions multiples et diversifiées sur divers marchés DeFi, le coût des emprunts et des retraits augmente car les smart contracts doivent effectuer des vérifications supplémentaires pour maintenir la liquidité et la solvabilité du système. Cela est dû à la nécessité de vérifier chaque position contre les exigences de collatéralisation et de santé financière du protocole pour prévenir le risque de défaut et assurer la stabilité de la plateforme.

2.3.4 Inéfficacité du modèle "pool"

Ce qu'il est intéressant d'observer ici c'est que, de par sa construction, la pool est inefficace. Une partie du capital reste inutilisée au sein de la pool pour garantir la liquidité [22], ce qui est grandement reflété dans les taux d'intérêts, puisqu'il y a un écart important entre le taux de prêt et d'emprunt. De plus, ces taux d'intérêts sont fixés par la gouvernance des projets, et ne reflètent pas forcément le besoin des prêteurs et emprunteurs.

Un modèle plus efficace serait d'obtenir un prêt pair à pair. En d'autres termes, si je peux prêter à un utilisateur, alors tous ses intérêts payés me reviendraient, ce qui donnerait lieu à une pool d'un seul utilisateur, ayant une utilisation de 100% et qui serait donc optimale pour le prêteur. L'emprunteur paierait toujours autant, mais les marchés seraient beaucoup plus efficaces.

C'est ce qu'a tenté de faire Aave avant de se replier sur un modèle de Pool avec EthLend [5].

EthLend était un marché décentralisé pour les prêts et emprunts de cryptomonnaies, où

les utilisateurs pouvaient emprunter des actifs en mettant en gage d'autres cryptomonnaies comme collatéral. Le fonctionnement de EthLend était basé sur des contrats intelligents sur la blockchain Ethereum, permettant des transactions transparentes sans intermédiaires.

- **Fonctionnement d'EthLend :** EthLend permettait aux emprunteurs de créer des demandes de prêt en spécifiant le montant, le collatéral offert, et la durée du prêt. Les prêteurs pouvaient alors parcourir ces demandes et financer celles qui leur semblaient sûres et rentables. Si un emprunteur ne remboursait pas le prêt, le prêteur pouvait liquider le collatéral pour récupérer ses fonds.
- **Raisons de la transition vers Aave :** Bien que le concept de EthLend était innovant, plusieurs défis ont mené à sa transition vers Aave. Voici les principales raisons :
  1. **Manque de liquidité :** EthLend fonctionnait sur un modèle de carnet d'ordres, où la correspondance directe entre offre et demande pouvait entraîner un manque de liquidité, rendant difficile la réalisation rapide des prêts et emprunts.
  2. **Complexité pour les utilisateurs :** La nécessité pour les emprunteurs et les prêteurs de s'accorder sur des termes spécifiques pour chaque prêt rendait le processus compliqué et moins attrayant pour les utilisateurs occasionnels.
  3. **Inefficacité du modèle de prêt pair-à-pair :** Le modèle p2p nécessitait une coordination constante entre les parties, ce qui pouvait être coûteux en termes de frais de transaction et de temps.
  4. **Problèmes d'évolutivité :** La plateforme avait du mal à gérer un grand nombre de prêts et de collatéraux, ce qui limitait son potentiel de croissance.
  5. **Concurrence et innovation :** Avec l'émergence de nouveaux protocoles offrant de meilleures solutions en termes de liquidité et d'efficacité, EthLend devait évoluer pour rester compétitif.

La transition vers Aave a marqué une évolution significative du modèle de prêt de EthLend, passant d'un carnet d'ordres à un modèle de pool de liquidité. Cette mutation a adressé plusieurs des problèmes mentionnés ci-dessus, en offrant notamment une liquidité instantanée grâce à des pools de prêts et en simplifiant l'expérience utilisateur.

Depuis cet échec, aucun autre projet n'a tenté l'approche décentralisée et pair à pair. C'est dans cet état de l'art que Morpho [6] est apparu.

Conclusion

Dans cette première partie, nous avons exploré le fonctionnement des indices financiers appliqués aux pools de liquidités sur des plateformes de finance décentralisée (DeFi). L'objectif

principal de ces indices est de refléter l'évolution des intérêts générés au sein d'une pool de manière à ce que tous les participants puissent en bénéficier équitablement et en temps réel.

Nous avons introduit la notion d'indice du marché $\lambda_\theta^S(t)$ qui ajuste la valeur nominale des dépôts en fonction des intérêts accumulés. Ce mécanisme assure que chaque interaction avec la pool met à jour l'indice et, par conséquent, l'intérêt accumulé est instantanément crédité aux balances des utilisateurs.

Nous avons souligné les défis liés à l'efficacité des pools de liquidités. En effet, une partie du capital reste statique pour maintenir la liquidité nécessaire, influant sur les taux d'intérêt et manifestant un écart notable entre les taux de prêt et d'emprunt. Cette inefficacité, accentuée par des paramètres de risques souvent fixés par la gouvernance des projets, ne reflète pas toujours les véritables besoins du marché.

Nous avons ensuite considéré l'approche de prêt pair à pair comme une alternative plus efficiente. Cette méthode permettrait une utilisation optimale du capital avec des intérêts entièrement reversés au prêteur. Cependant, malgré sa potentialité, cette approche a rencontré des obstacles pratiques, comme illustré par le pivot d'EthLend vers Aave, mettant en lumière les difficultés de mise à l'échelle et de flexibilité requises pour un protocole de DeFi.

Enfin, nous avons observé que, malgré les tentatives précédentes et les défis associés, aucun projet n'a réussi à mettre en œuvre une solution décentralisée et pair à pair viable depuis Aave. C'est dans ce contexte que le projet Morpho a vu le jour, visant à combler cette lacune dans l'écosystème DeFi.

Cette première partie met en évidence les complexités et les opportunités inhérentes à la DeFi, et sert de fondement à la discussion sur les innovations et les améliorations possibles dans le domaine des pools de liquidités et des protocoles de prêt.

CHAPITRE 3 GÉNÉRALITÉS SUR MORPHO

Morpho [1] représente une surcouche pour les protocoles de liquidité basés sur des pools, qui libère et optimise ses taux d'utilisation tout en maintenant pour l'utilisateur la même liquidité et les mêmes risques de liquidation que ceux offerts par la pool originale. Il agit en tant qu'intermédiaire entre l'utilisateur et la pool, favorisant le jumelage en pair-à-pair des prêteurs et des emprunteurs lorsque cela est possible.

3.0.1 Fonctionnement global

Dans la pratique, Morpho propose une expérience utilisateur identique à celle de Aave [4] ou Compound [3]. Les utilisateurs peuvent déposer ou emprunter des fonds et les retirer à tout moment. Les procédures de liquidation sont calquées sur celles de la pool sous-jacente. La différence notable pour l'utilisateur est la possibilité d'obtenir de meilleurs taux, que ce soit en tant que prêteur ou emprunteur.

Lorsque les conditions le permettent, Morpho met en relation directe un prêteur et un emprunteur en utilisant un algoritheoreme de correspondance. Le dépôt du prêteur est alors retiré de la pool et la dette de l'emprunteur est remboursée. Dès cet instant, les taux ne sont plus contraints par ceux de la pool.

En établissant un taux intermédiaire entre le taux de prêt et le taux d'emprunt de la pool (par exemple $\frac{r^S+r^B}{2}$), les deux parties bénéficient d'une meilleure offre.

Si la mise en relation pair-à-pair n'est pas réalisable, les lignes de crédit sont réintégrées dans le protocole de liquidité sous-jacent, permettant ainsi aux utilisateurs de profiter des taux standards de la pool. Cette flexibilité garantit l'efficience du pair-à-pair tout en préservant la liquidité inhérente aux protocoles basés sur des pools.

Exemple 3.0.1.1. *Illustrons avec un exemple simple. Sur Aave, les taux d'intérêt sont de* $r^S = 1\%$ *pour le prêt et de* $r^B = 5\%$ *pour l'emprunt.*

- *À l'instant* t_0*, Alice dépose 10 ETH sur Morpho. En l'absence d'emprunteurs sur Morpho, ses fonds sont placés sur Aave, lui rapportant ainsi* 1% *d'intérêt annuel.*
- *Au temps* t_1*, Bob souhaite emprunter 10 ETH. Morpho retire les 10 ETH d'Alice de Aave et les prête à Bob.*

Suite à cette transaction, Alice et Bob bénéficient tous deux d'un taux d'intérêt de 3%*, situation avantageuse pour chacun d'eux.*

1. Initialement nommé Optimizers dans sa première version.

FIGURE 3.1 Fonctionnement de Morpho pour les prêteurs.

FIGURE 3.2 Fonctionnement de Morpho pour les emprunteurs.

Nous allons à présent examiner les détails d'implémentation, en analysant notamment la manière dont Morpho réalise la mise en pair-à-pair.

3.0.2 Les balances des utilisateurs

Pour suivre la position de chaque utilisateur sur Morpho, il est nécessaire de définir la manière dont ces informations seront enregistrées sur la blockchain Ethereum.

Balance des prêteurs : Les prêteurs disposent de deux types de balances dans Morpho :

- Les dépôts dans la pool s^{Pool}, qui sont mesurés dans une unité s'accroissant au taux r^S, représentent le montant déposé par un prêteur dans la pool. Pour obtenir la valeur comprenant les intérêts, on multiplie cette balance par l'indice des intérêts de la pool λ_θ^S.
- La portion en pair-à-pair s^{P2P}, qui croît au taux r^{P2P}, représente la partie du dépôt de l'utilisateur bénéficiant des taux avantageux du marché en pair-à-pair. La valeur totale, intérêts inclus, est obtenue en multipliant cette balance par l'indice des intérêts du marché en pair-à-pair $\lambda_\theta^{P2P}(t)$ (la fonctionnalité de cet indice sera détaillée ultérieurement).

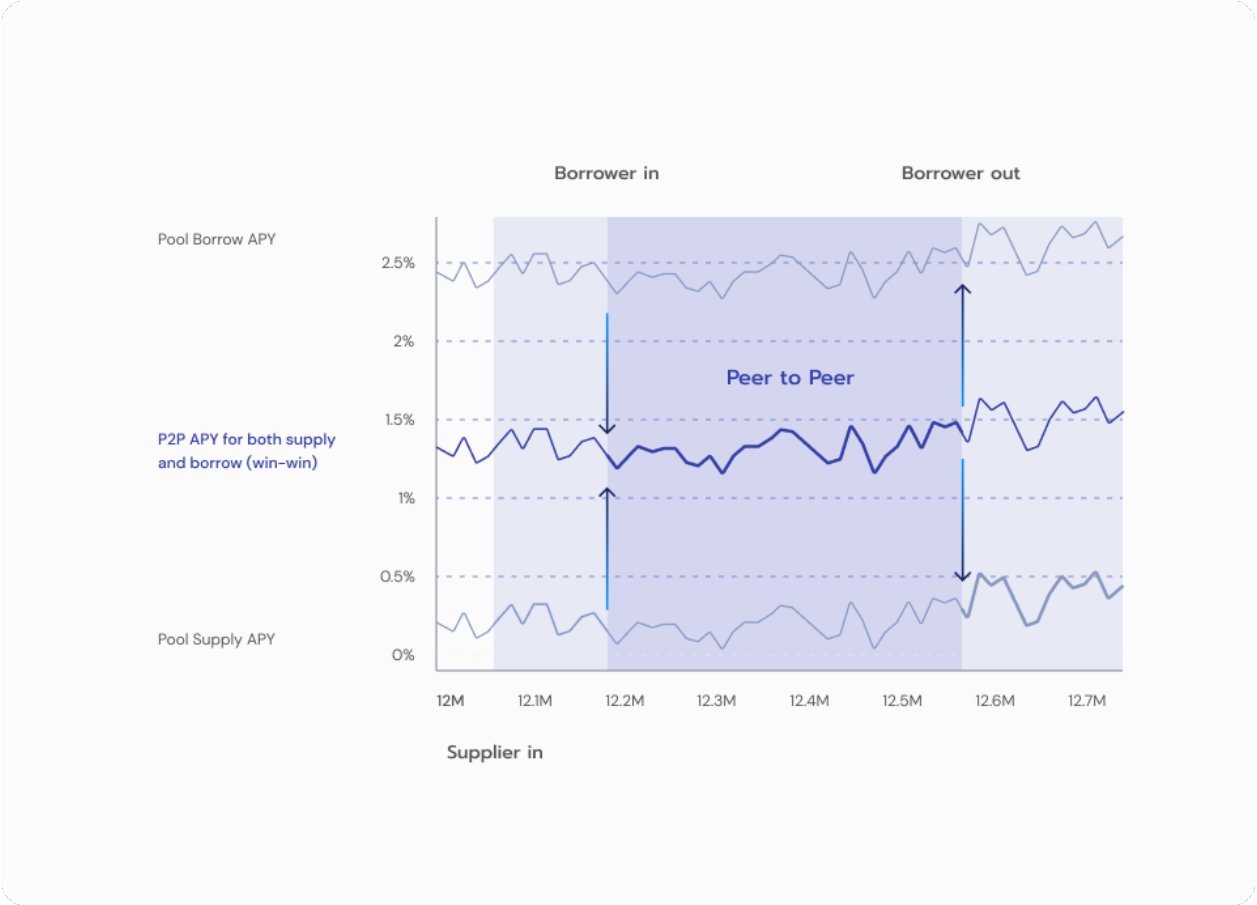


FIGURE 3.3 Taux en pair-à-pair de Morpho.

La séparation des balances est le mécanisme qui confère à Morpho ses propriétés de non-fongibilité [2]. Le dépôt total d'un utilisateur est l'agrégat des deux soldes, qui évoluent à deux taux distincts.

Nous définissons le dépôt total de l'utilisateur u au temps θ par la notation suivante :

$$\omega_\theta^S(u) = s_\theta^{Pool}(u) \cdot \lambda_\theta^S + s_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} \tag{3.1}$$

Balance des emprunteurs : Le même principe est appliqué aux soldes des emprunteurs. La dette envers la pool b^{Pool} représente l'endettement de l'utilisateur sur la pool, croissant à un taux r^B et actualisé avec l'indice λ_θ^B. Parallèlement, le solde d'emprunt en pair à pair b^{P2P} augmente à un taux r^{P2P}, utilisant le même indice que celui des prêteurs en pair à pair λ_θ^{P2P}.

2. Cela signifie qu'un utilisateur ne peut pas transférer une portion de sa position à quelqu'un d'autre, ce qui est possible dans une pool de liquidité.

Le total emprunté par un utilisateur est la somme de ces deux soldes, ramenée à l'unité de base du token. Nous utilisons la notation suivante pour le solde total emprunté par l'utilisateur u au temps θ :

$$\omega_\theta^B(u) = b_\theta^{Pool}(u) \cdot \lambda_\theta^B + b_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} \tag{3.2}$$

3.0.3 Calcul des taux d'intérêt pour les utilisateurs

Morpho définit quatre taux d'intérêt : pour les prêteurs et les emprunteurs sur la pool, les taux sont identiques à ceux proposés par la pool. Pour les utilisateurs en pair à pair, un taux spécifique est appliqué, indépendant des contraintes de la pool. Une différence mineure entre les taux d'un prêteur et d'un emprunteur en pair à pair peut être introduite par une commission pour le protocole Morpho (voir Section 2.6 du Livre blanc de Morpho [3]).

Les taux théoriques

Pour mieux comprendre comment les intérêts sont calculés, nous considérons d'abord des taux théoriques, définis comme le coût pour les emprunteurs (ou le gain pour les prêteurs) par unité de temps [4].

Les utilisateurs non appariés sont directement déposés sur la pool et les intérêts sont calculés par le protocole sous-jacent. Les taux r^B et r^S sont accessibles directement depuis la pool.

Il est important de noter que, aux yeux de la pool, Morpho est considéré comme un seul utilisateur. Le smart contract Morpho rajoute de la logique pour redistribuer les intérêts accumulés à travers le marché de la pool à tous les utilisateurs en attente dans la pool. Ainsi, même pour les utilisateurs sur la pool, il est nécessaire de comptabiliser les intérêts au sein du protocole Morpho pour redistribuer les intérêts que Morpho a accumulés.

Le Spread Pour les utilisateurs appariés, Morpho a la liberté de fixer un taux d'intérêt entre r^B et r^S, un espace que nous appelons *le spread*. Dans sa version initiale, Morpho a choisi de calculer la moyenne aritheoremétique pondérée des taux de prêt et d'emprunt de la pool, ajustée par un paramètre α situé entre 0 et 1, appelé le curseur pair à pair.

3. Ajouter la référence pertinente ici
4. Annuellement pour Aave, par bloc pour Compound

Le taux en pair à pair est alors donné par :

$$\begin{aligned} r^{\alpha} &= r^{S} + \alpha(r^{B} - r^{S}) \\ &= (1-\alpha)r^{S} + \alpha r^{B} \end{aligned}$$

Par exemple, pour le “taux médian" avec $\alpha = \frac{1}{2}$:

$$r^{\alpha} = r^{\frac{1}{2}} = \frac{r^{S} + r^{B}}{2}$$

Ce taux est appliqué autant aux prêteurs qu'aux emprunteurs en pair à pair, réduisant à zéro le spread pour ces utilisateurs et optimisant l'efficacité du marché.

Frais d'utilisation Cependant, Morpho se réserve le droit de prélever des frais sur ces taux [5], afin de soutenir son développement à long terme. Ces frais recréent un écart (spread) au sein de l'écart initial, mais tout en assurant une amélioration de Pareto, dans le sens où, si Morpho choisissait de prélever des frais de 100%, les utilisateurs en pair à pair se verraient simplement appliquer le taux de la pool. Ces frais ne s'appliquent qu'aux utilisateurs en pair à pair et n'affectent pas les utilisateurs en file d'attente sur la pool qui bénéficient de l'expérience offerte par le protocole sous-jacent. Soit ρ le *facteur de réserve* de Morpho, représentant la part du spread qui est allouée au protocole.

Les taux ajustés sont alors :

$$\begin{aligned} r^{\rho^S} &= r^{\alpha} - \rho(r^{\alpha} - r^{S}) \\ &= (1-\rho)r^{\alpha} + \rho r^{S} \\ &= (1-\rho)((1-\alpha)r^{S} + \alpha r^{B}) + \rho r^{S} \\ &= (1-\alpha+\alpha\rho)r^{S} + \alpha(1-\rho)r^{B} \end{aligned}$$

$$\begin{aligned} r^{\rho^B} &= r^{\alpha} + \rho(r^{B} - r^{\alpha}) \\ &= (1-\rho)r^{\alpha} + \rho r^{B} \\ &= (1-\rho)((1-\alpha)r^{S} + \alpha r^{B}) + \rho r^{B} \\ &= (1-\alpha)\rho r^{S} + (1-\rho+\alpha\rho)r^{B} \end{aligned}$$

5. Le taux des frais est déterminé par la gouvernance et est actuellement fixé à 0% pour tous les protocoles de Morpho.

La différence $r^{\rho^B} - r^{\rho^S} = \rho(r^B - r^S)$ est le pourcentage ρ de l'écart initial $r^B - r^S$.

Limitation Ces taux ne sont que théoriques, car Morpho ne peut pas intervenir directement sur les taux de la pool. Comme mentionné précédemment, les taux d'intérêt sur la pool dépendent du taux d'utilisation et sont donc ajustés à chaque transaction avec la pool. Les interactions de Morpho avec la pool étant incluses dans toutes les interactions avec celle-ci, Morpho ne peut pas ajuster ses taux p2p à chaque mise à jour des taux de la pool. Cela est dû au fait que la blockchain ne se met à jour que lorsqu'une transaction est effectuée. Par conséquent, une transaction devrait être initiée sur Morpho pour actualiser les taux p2p (et par conséquent les indices p2p) à chaque interaction avec la pool, ce qui serait indésirable et onéreux.

Pour surmonter ce défi, nous utiliserons une méthode basée sur une approximation des indices.

Calcul des indices sur Morpho

Définissons les coefficients β^S et β^B, avec $\beta^B + \beta^S = 1$, de sorte que le taux Morpho r soit $\beta^S r^S + \beta^B r^B$. Ces coefficients déterminent la position du taux Morpho dans le spread. Par exemple, si β^S est proche de 1, alors r sera proche du taux r^S de la pool. Soit λ l'indice associé à r, ce qui nous permet de généraliser les résultats de cette sous-section. En particulier, nous pouvons appliquer le théorème 3.0.3.1 aux taux r^α, r^{ρ^S}, r^{ρ^B} et également aux taux r^{γ^S} et r^{γ^B}.

Comme mentionné précédemment, Morpho n'utilise pas les taux de la pool pour mettre à jour ses indices, mais s'appuie plutôt sur l'évolution des indices de la pool. Le théorème suivant présente la formule utilisée pour actualiser les indices de Morpho de cette façon. Notez que les taux de la pool ne figurent pas dans la formule énoncée par le théorème.

Nous supposons que la pool sous-jacente capitalise les intérêts pour mettre à jour ses indices.

Théorème 3.0.3.1. *Soient $\beta^S \in \mathbb{R}_+$ et $\beta^B \in \mathbb{R}_+$ tels que $\beta^S + \beta^B = 1$, et soit $(\lambda_i)_{i\in\mathbb{N}}$ une séquence telle que :*

$$\forall t \in \mathbb{N}, \forall t' \in \mathbb{N}, t > t' \Rightarrow \lambda_t = \lambda_{t'} \prod_{t' \leq i < t} (1 + \beta^S r_i^S + \beta^B r_i^B)$$

Alors, avec $R_{t',t}$ le taux maximum r_i^B pour $i \in [\![t', t[\![$:

$$\forall t \in \mathbb{N}, \forall t' \in \mathbb{N}, t > t' \Rightarrow \lambda_t = \lambda_{t'} \left(\beta^S \frac{\lambda_t^S}{\lambda_{t'}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{t'}^B} + O(R_{t',t}^2) \right)$$

Démonstration. Soit $t, t' \in \mathbb{N}$ avec $t' < t$. L'exactitude de l'approximation donnée par le théorème dépend de la petitesse des taux, impliquant que le maximum de ces taux $R_{t',t}$ est également petit.

$$\begin{aligned}
\lambda_t &= \lambda_{t'} \prod_{t' \leq i < t} (1 + \beta^S r_i^S + \beta^B r_i^B) \\
&= \lambda_{t'} \left(1 + \sum_{t' \leq i < t} (\beta^S r_i^S + \beta^B r_i^B) + O(R_{t',t}^2) \right) \\
&= \lambda_{t'} \left(1 + \beta^S \sum_{t' \leq i < t} r_i^S + \beta^B \sum_{t' \leq i < t} r_i^B + O(R_{t',t}^2) \right)
\end{aligned}$$

D'autre part, pour les indices de la pool, nous avons :

$$\begin{aligned}
\frac{\lambda_t^S}{\lambda_{t'}^S} &= \prod_{t' \leq i < t} (1 + r_i^S) \\
&= 1 + \sum_{t' \leq i < t} r_i^S + O(R_{t',t}^2)
\end{aligned}$$

et de manière similaire,

$$\begin{aligned}
\frac{\lambda_t^B}{\lambda_{t'}^B} &= \prod_{t' \leq i < t} (1 + r_i^B) \\
&= 1 + \sum_{t' \leq i < t} r_i^B + O(R_{t',t}^2)
\end{aligned}$$

En combinant ces résultats, nous obtenons :

$$\begin{aligned}
\lambda_t &= \lambda_{t'} \left(1 + \beta^S (\frac{\lambda_t^S}{\lambda_{t'}^S} - 1) + \beta^B (\frac{\lambda_t^B}{\lambda_{t'}^B} - 1) + O(R_{t',t}^2) \right) \\
&= \lambda_{t'} \left(\beta^S \frac{\lambda_t^S}{\lambda_{t'}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{t'}^B} - (\beta^S + \beta^B) + 1 + O(R_{t',t}^2) \right) \\
&= \lambda_{t'} \left(\beta^S \frac{\lambda_t^S}{\lambda_{t'}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{t'}^B} + O(R_{t',t}^2) \right)
\end{aligned}$$

□

Si la pool sous-jacente utilise un intérêt simple, l'approximation donnée par le théorème devient un calcul exact. Ce résultat est d'autant plus remarquable qu'il peut être calculé en temps constant ; c'est-à-dire, un calcul en complexité $O(1)$ suffit pour mettre à jour les

indices p2p. Effectivement, l'indice de la pool encapsule l'historique intégral de tout ce qui a été distribué depuis sa dernière mise à jour.

Expliquons maintenant comment les indices sont mis à jour dans Morpho en commençant par le cas le plus simple, sans facteur de réserve.

Sans facteur de réserve Soit $(u_i)_{i\in\mathbb{N}}$ la séquence des instants de mise à jour de l'indice sur Morpho, et soient $(\lambda_i^S)_{i\in\mathbb{N}}$ et $(\lambda_i^B)_{i\in\mathbb{N}}$ les indices de supply et de borrow sur la pool, respectivement. Introduisons $(\lambda_i^\alpha)_{i\in\mathbb{N}}$ pour les indices Morpho, en tenant compte du paramètre α.

Pour $t, t' \in \mathbb{N}$ avec $t' < t$, nous avons :

$$\begin{aligned}\lambda_t^\alpha &= \lambda_{t'}^\alpha \prod_{t'\le i<t} (1 + r_i^\alpha) \\ &= \lambda_{t'}^\alpha \prod_{t'\le i<t} (1 + (1-\alpha) r_i^S + \alpha r_i^B)\end{aligned}$$

En utilisant le théorème 3.0.3.1 avec $\beta^S = (1-\alpha)$ et $\beta^B = \alpha$:

$$\begin{aligned}\lambda_t^\alpha &= \lambda_{t'}^\alpha \left((1-\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + \alpha \frac{\lambda_t^B}{\lambda_{t'}^B} + O(R_{t',t}^2) \right) \\ &\approx \lambda_{t'}^\alpha \left((1-\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + \alpha \frac{\lambda_t^B}{\lambda_{t'}^B} \right)\end{aligned}$$

Cette dernière expression est utilisée pour actualiser les indices sans considérer de facteur de réserve.

Avec facteur de réserve : Nous intégrons maintenant le facteur de réserve. Comme mentionné précédemment, nous souhaitons que les tarifs appliqués par les utilisateurs soient :

$$\begin{aligned}r^{\rho^S} &= (1 - \alpha + \rho\alpha) r^S + (\alpha - \rho\alpha) r^B, \\ r^{\rho^B} &= (1 - \rho - \alpha + \rho\alpha) r^S + (\rho + \alpha - \rho\alpha) r^B.\end{aligned}$$

Nous désignons les indices correspondants par $(\lambda_i^{\rho^S})_{i\in\mathbb{N}}$ et $(\lambda_i^{\rho^B})_{i\in\mathbb{N}}$. Cette notation permet de prendre en compte le facteur de réserve, en complément du curseur pair à pair.

Soit $t \in \mathbb{N}$ et $t' \in \mathbb{N}$ avec $t' < t$. Nous avons :

$$\begin{aligned}\lambda_t^{\rho^S} &= \lambda_{t'}^{\rho^S} \prod_{t' \leq i < t} (1 + r_i^{\rho^S}), \\ &= \lambda_{t'}^{\rho^S} \prod_{t' \leq i < t} \left(1 + (1 - \alpha + \rho\alpha) r_i^S + (\alpha - \rho\alpha) r_i^B\right).\end{aligned}$$

En appliquant le théorème 3.0.3.1 avec $\beta^S = (1 - \alpha + \rho\alpha)$ et $\beta^B = (\alpha - \rho\alpha)$, nous obtenons :

$$\lambda_t^{\rho^S} = \lambda_{t'}^{\rho^S} \left((1 - \alpha + \rho\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + (\alpha - \rho\alpha)\frac{\lambda_t^B}{\lambda_{t'}^B} + O\left((R_{t',t})^2\right)\right). \tag{3.3}$$

De même, pour les indices d'emprunt :

$$\lambda_t^{\rho^B} = \lambda_{t'}^{\rho^B} \left((1 - \rho - \alpha + \rho\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + (\rho + \alpha - \rho\alpha)\frac{\lambda_t^B}{\lambda_{t'}^B} + O\left((R_{t',t})^2\right)\right). \tag{3.4}$$

Avec le facteur de réserve, ces résultats nous permettent de mettre à jour les indices. Ces formules offrent une méthode de calcul efficace adaptée à une implémentation algoritheoremique.

3.0.4 Inégalité entre pairs

Le lemme d'*intégrité pair à pair* ne s'applique plus en présence du facteur de réserve. Cependant, il est toujours possible de démontrer que la somme des soldes pair à pair des prêteurs équivaut à la somme des soldes pair à pair des emprunteurs, augmentée de la réserve impayée, notée P :

Lemme 3.0.4.1. *Pour tout $\theta \in \Theta$, on a*

$$\sum_u s^{P2P}(u) \cdot \lambda_\theta^{\rho^S} + P_\theta = \sum_u b^{P2P}(u) \cdot \lambda_\theta^{\rho^B}.$$

Nous en déduisons directement le corollaire suivant :

Lemme 3.0.4.2 (Inégalité entre pairs)**.** *Pour tout $\theta \in \Theta$,*

$$\sum_u s^{P2P}(u) \cdot \lambda_\theta^{\rho^S} \leq \sum_u b^{P2P}(u) \cdot \lambda_\theta^{\rho^B}.$$

Ces résultats sont suffisants pour établir le théorème de non-liquidation (3.1.2.1).

3.0.5 Calculs de taux et amélioration de Pareto

Cette section démontre que les taux proposés par Morpho sont supérieurs à ceux du marché sous-jacent. Nous partons du principe que les taux initiaux de l'indice pair à pair se situent entre les taux de l'indice d'offre λ^S et l'indice d'emprunt λ^B. Nous montrons ensuite que les taux de l'indice demeurent dans cette fourchette. Un calcul exact des taux est présenté, ce qui aboutit à une estimation des taux qui est à la fois plus intelligible et plus facile à calculer.

Considérons (λ_i) une séquence qui ne varie qu'aux instants (u_n), ces derniers représentant les moments de mise à jour. Ainsi, pour $t = u_n$, on a :

$$\lambda_t = \lambda_{u_{n-1}} \left(\beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B} \right),$$

avec $\beta^S + \beta^B = 1$. Cette propriété est valable pour (λ_i^α), $(\lambda_i^{\rho^S})$, $(\lambda_i^{\rho^B})$, ainsi que pour (λ_i^S) et (λ_i^B). Rappelons que les coefficients β^S et β^B sont dérivés du curseur pair à pair α et du facteur de réserve ρ de Morpho.

Amélioration des taux d'indexation

Théorème 3.0.5.1. *L'indice λ_i reste dans l'intervalle $[\lambda_i^S, \lambda_i^B]$.*

Démonstration. Nous démontrons ce théorème par récurrence. On suppose qu'elle est vraie à 0, il suffit donc de prouver qu'elle est vraie à $t = u_n$, sachant que :

$$\lambda_{u_{n-1}}^S \leq \lambda_{u_{n-1}} \leq \lambda_{u_{n-1}}^B$$

Comme le taux d'offre est inférieur au taux d'emprunt, on a

$$\frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} \leq \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$$

La moyenne de ces taux devrait se situer au milieu, donc :

$$\frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} \leq \beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B} \leq \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$$

En utilisant l'hypothèse de récurrence, on obtient

$$\lambda_{u_{n-1}}^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} \leq \lambda_{u_{n-1}} (\beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}) \leq \lambda_{u_{n-1}}^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$$

c'est ce que nous avons voulu prouver :

$$\lambda_t^S \leq \lambda_t \leq \lambda_t^B$$

□

Amélioration des taux

Soit u_{n-1} la dernière fois que les taux d'indexation ont été mis à jour. Avec $\Delta t = t - u_{n-1}$, soit $\bar{r}_t^S$ le taux tel que :

$$(1 + \bar{r}_t^S)^{\Delta t} = \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S}$$

Le taux $\bar{r}_t^S$ est formellement défini par :

$$\bar{r}_t^S = \left(\frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} \right)^{1/\Delta t} - 1$$

Le taux $\bar{r}_t^S$ est une "moyenne", ce qui signifie qu'il s'agit du taux auquel le taux de l'indice λ^S augmenterait si son taux était constant entre u_{n-1} et t. En d'autres termes, on peut considérer qu'un fournisseur a bénéficié de ce taux moyen sur toute la période Δt. Nous définissons $\bar{r}_t^S$ et $\bar{r}_t$ de la même manière :

$$(1 + \bar{r}_t^B)^{\Delta t} = \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$$

$$(1 + \bar{r}_t)^{\Delta t} = \beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$$

Ainsi, le taux rencontré par les utilisateurs pair à pair de Morpho entre les temps de mise à jour u_{n-1} et $t = u_n$ est donné par :

$$\bar{r}_t = (\beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B})^{1/\Delta t} - 1$$

Théorème 3.0.5.2. *Le taux constaté $\bar{r}_t$ se situe dans l'intervalle $[\bar{r}_t^S, \bar{r}_t^B]$.*

Démonstration. Définissons la fonction $f : \mathbb{R}^+ \to \mathbb{R}$ telle que pour tout $x \geq 0$, $f(x) = (1 + x)^{1/\Delta t} - 1$. Supposons que

$$\frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} \leq \beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B} \leq \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$$

et comme f est croissante, il en découle que

$$f\left(\frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} - 1\right) \leq f\left(\beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B} - 1\right) \leq f\left(\frac{\lambda_t^B}{\lambda_{u_{n-1}}^B} - 1\right)$$

d'où

$$\bar{r}_t^S \leq \bar{r}_t \leq \bar{r}_t^B.$$

Cela démontre que le taux Morpho est mieux positionné que le taux de la pool.

□

Calcul et estimation du taux de Morpho

Pour déterminer la position du taux moyen entre le taux d'offre et le taux d'emprunt, calculons β_r^S et β_r^B tels que $\beta_r^S + \beta_r^B = 1$ et $\bar{r}_t = \beta_r^S \bar{r}_t^S + \beta_r^B \bar{r}_t^B$. Le calcul exact est donné par :

$$\beta_r^S = \frac{\bar{r}_t - \bar{r}_t^B}{\bar{r}_t^S - \bar{r}_t^B}$$

et

$$\beta_r^B = 1 - \beta_r^S.$$

Ainsi, tout utilisateur ayant accès aux taux de la pool sous-jacente peut déterminer la position du taux Morpho entre le taux d'offre et le taux d'emprunt.

Lorsque les quantités $\frac{\lambda_t^S}{\lambda_{u_{n-1}}^S}$ et $\frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}$ sont faibles, nous pouvons utiliser l'approximation suivante de la fonction f pour les petites valeurs de x :

$$\begin{aligned} f(x) &\approx f(0) + f'(0) \cdot x \\ &\approx \Delta t \cdot x, \end{aligned}$$

et par conséquent :

$$\begin{aligned} \beta_r^S &\approx \frac{\Delta t \cdot \left(\beta^S \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} + \beta^B \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}\right) - \Delta t \cdot \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}}{\Delta t \cdot \frac{\lambda_t^S}{\lambda_{u_{n-1}}^S} - \Delta t \cdot \frac{\lambda_t^B}{\lambda_{u_{n-1}}^B}} \\ &\approx \beta^S. \end{aligned}$$

Il s'ensuit que $\beta_r^B \approx \beta^B$, une approximation indépendante du temps, qui fournit une règle empirique simple pour estimer la position du taux Morpho.

Pour le taux de prêt pair à pair, prenant en compte le facteur de réserve, nous avons $\beta^B = \alpha - \rho\alpha$. Cela est utile pour calculer le taux effectif pour un utilisateur ayant des positions à la fois dans la pool et en pair à pair, où le taux subi est une moyenne pondérée du taux pair à pair et du taux de la pool.

3.1 Liquidation

3.1.1 Description

La liquidation sur Morpho s'opère de manière analogue à celle sur la pool sous-jacente. Lorsqu'une position d'un utilisateur devient liquidable, un liquidateur peut rembourser une partie de la dette de l'utilisateur et récupérer une portion correspondante de son collatéral. Morpho utilise les mêmes paramètres pour les facteurs de collatéral (`closeFactor`) et le `bonus de liquidation` que ceux en vigueur dans la pool sous-jacente, ainsi que le même oracle pour évaluer les prix des actifs empruntés et mis en collatéral. La distinction principale par rapport à la pool concerne le calcul du montant des actifs fournis et empruntés ; pour chaque utilisateur, il est nécessaire de prendre en compte les soldes sur la pool et en pair à pair (voir Section 3.1).

3.1.2 Intégrité de Morpho sur la Pool

Comme nous l'avons évoqué, pour la pool sous-jacente, Morpho représente l'ensemble des utilisateurs dont les positions n'ont pas été appariées en pair à pair. Autrement dit, Morpho est considéré comme un utilisateur unique de la pool, disposant de son propre facteur de santé. Le théorème de non-liquidation est crucial pour le fonctionnement de Morpho ; il assure que la position globale de Morpho sur la pool ne peut être liquidée.

Illustrons les positions au sein de Morpho à l'aide du graphique de la Figure 3.4. La barre de gauche indique la totalité des actifs fournis via Morpho et la barre de droite, ceux empruntés, exprimés dans une même unité de mesure arbitraire. Pour chaque barre, la partie inférieure représente la liquidité correspondante sur la pool sous-jacente, celle qui n'a pas été appariée en pair à pair. Notons que la portion compensée de l'offre est identique à celle de l'emprunt.

Le théorème de non-liquidation postule que la position globale de Morpho vis-à-vis de la pool sous-jacente ne peut être liquidée.

Cette hypothèse se base sur l'idée que les positions sûres sont celles pour lesquelles la valeur des actifs empruntés est une fraction de celle des actifs fournis. Puisque Morpho contraint ses utilisateurs à maintenir des positions sûres (en tenant compte à la fois de leurs actifs dans

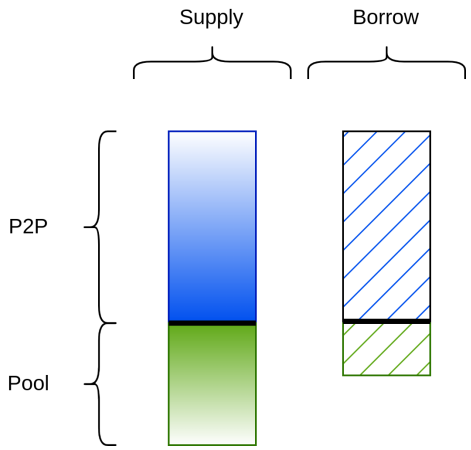


FIGURE 3.4 Représentation schématique des positions sur Morpho.

la pool et de ceux en pair à pair) par le biais du mécanisme de liquidation, il s'ensuit que la somme des emprunts sur la pool et en pair à pair constitue une fraction de la somme des offres sur la pool et en pair à pair.

Sur le graphique, cette relation est représentée par la proportion de la barre d'emprunt par rapport à la barre de prêt dans leur intégralité.

La suppression des parties correspondantes en emprunt et en offre ne fait que diminuer cette fraction, ce qui revient à comparer les emprunts sur la pool avec les offres sur la pool, traduisant ainsi la position de Morpho en tant qu'entité unique sur la pool.

Le graphique illustre cette idée en comparant la partie inférieure de la barre d'emprunt à la partie inférieure de la barre de prêt.

Nous démontrerons tout d'abord le lemme *intégrité de la position sur la pool* et le lemme *intégrité en pair à pair*, qui sont fondamentaux pour établir le théorème de non-liquidation. Rappelons que b_θ et s_θ désignent respectivement les soldes d'emprunt et de prêt sur la pool.

Intégrité de la position de la pool

Lemme 3.1.2.1 (Intégrité de la position de la pool)**.** *La position de Morpho sur la pool est toujours équivalente à la somme des positions individuelles de tous ses utilisateurs sur cette même pool.*

$$\forall \theta \in \Theta_{M_\psi}, \begin{cases} s_\theta(\overline{M_\psi}) = \sum_u s_\theta^{Pool}(u), \\ b_\theta(\overline{M_\psi}) = \sum_u b_\theta^{Pool}(u). \end{cases}$$

Démonstration. Nous procédons par induction.

Cas de base : Initialement, il n'y a aucun utilisateur sur Morpho :

$$\sum_u s^{\text{Pool}}(u) = \sum_u b^{\text{Pool}}(u) = 0,$$

et Morpho n'a aucune position sur la pool :

$$s(\overline{M_\psi}) = b(\overline{M_\psi}) = 0.$$

Ainsi, les égalités sont vérifiées dans l'état initial.

Hypothèse d'induction : Supposons que l'équilibre est maintenu pour un état $\sigma_i \in \Sigma$. Considérons une transition vers un état $\sigma_{i+1} \in \Sigma$ suite à un appel de fonction, par exemple `supply`. Soit $d = (\theta, u, x) \in D_{\texttt{supply}}$ l'entrée pour cette transition. Pour simplifier, nous nous limitons au marché θ puisque les autres ne sont pas affectés, ce qui nous permet de négliger l'indice θ.

Nous utilisons λ^S et λ^B pour désigner respectivement $\sigma_{i+1}(\lambda^S)$ et $\sigma_{i+1}(\lambda^B)$.

Considérons d'abord l'**appariement pair-à-pair**. Soit x^{μ_B} la quantité totale correspondue par `matchBorrowers` (noté `pair_a_pair` dans le pseudocode). La quantité totale d'emprunts sur la pool diminue de x^{μ_B}/λ^B, et la position d'emprunt de Morpho sur la pool diminue de x^{μ_B}/λ^B.

$$\sum_u \sigma_{i+1}(b^{\text{Pool}}(u)) = \sum_u \sigma_i(b^{\text{Pool}}(u)) - \frac{x^{\mu_B}}{\lambda^B},$$

$$\sigma_{i+1}(b(\overline{M_\psi})) = \sigma_i(b(\overline{M_\psi})) - \frac{x^{\mu_B}}{\lambda^B}.$$

Considérons ensuite le **fournissement à la pool**. Le montant $x^{\text{Pool}}/\lambda^S$ est ajouté au solde de fourniture de l'utilisateur avec $x^{\text{Pool}} = x - x^{\text{P2P}}$. Le solde de prêt de la pool augmente alors de $x^{\text{Pool}}/\lambda^S$:

$$\sum_u \sigma_{i+1}(s^{\text{Pool}}(u)) = \sum_u \sigma_i(s^{\text{Pool}}(u)) + \frac{x^{\text{Pool}}}{\lambda^S},$$

$$\sigma_{i+1}(s(\overline{M_\psi})) = \sigma_i(s(\overline{M_\psi})) + \frac{x^{\text{Pool}}}{\lambda^S}.$$

Puisque les égalités sont maintenues dans σ_i, elles le seront aussi dans σ_{i+1} :

$$\sigma_{i+1}(s(\overline{M_\psi})) = \sum_u \sigma_{i+1}(s^{\text{Pool}}(u)),$$

$$\sigma_{i+1}(b(\overline{M_\psi})) = \sum_u \sigma_{i+1}(b^{\text{Pool}}(u)).$$

La preuve est analogue pour les autres types de transitions. □

Intégrité pair-à-pair

Lemme 3.1.2.2 (Intégrité pair-à-pair)**.** *La somme des soldes pair-à-pair des fournisseurs est toujours égale à celle des emprunteurs.*

$$\forall \theta \in \Theta_{M_\psi}, \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} = \sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P}.$$

Démonstration. Nous adoptons une approche similaire à celle du lemme 3.1.2.1.

Cas de base : L'égalité est trivialement vérifiée à l'initialisation du système.

Hypothèse d'induction : Supposons que l'invariant est maintenu pour un état $\sigma_i \in \Sigma$ et montrons qu'il persiste suite à une transition consécutive à un appel de la fonction `supply`. Soit $d = (\theta, u, x) \in D_{\texttt{supply}}$ l'entrée pour cet appel.

L'égalité est préservée après la mise à jour de l'indice, donc pour simplifier, on écrit $\lambda_\theta^{\text{P2P}}$ pour $\sigma_{i+1}(\lambda_\theta^{\text{P2P}})$.

Examinons d'abord le **prêt pair-à-pair**. Soit x^{μ_B} la quantité appariée par $\texttt{matchBorrowers}_\theta$. Le solde pair-à-pair du fournisseur augmente de $x^{\mu_B}/\lambda_\theta^{\text{P2P}}$, de même que le total des soldes d'emprunt pair-à-pair.

$$\sum_u \sigma_{i+1}(s_\theta^{\text{P2P}}(u)) \cdot \lambda_\theta^{\text{P2P}} = \left(\sum_u \sigma_i(s_\theta^{\text{P2P}}(u)) + \frac{x^{\mu_B}}{\lambda_\theta^{\text{P2P}}}\right) \cdot \lambda_\theta^{\text{P2P}},$$

$$\sum_u \sigma_{i+1}(b_\theta^{\text{P2P}}(u)) \cdot \lambda_\theta^{\text{P2P}} = \left(\sum_u \sigma_i(b_\theta^{\text{P2P}}(u)) + \frac{x^{\mu_B}}{\lambda_\theta^{\text{P2P}}}\right) \cdot \lambda_\theta^{\text{P2P}}.$$

Notez que l'étape **fournissement à la pool** n'affecte pas les soldes en pair-à-pair. Ainsi, en considérant que l'équilibre était respecté dans σ_i, il en sera de même dans σ_{i+1} :

$$\sum_u \sigma_{i+1}(s_\theta^{\text{P2P}}(u)) \cdot \lambda_\theta^{\text{P2P}} = \sum_u \sigma_{i+1}(b_\theta^{\text{P2P}}(u)) \cdot \lambda_\theta^{\text{P2P}}.$$

La preuve est similaire pour les autres transitions. □

Théorème de non-liquidation

Passons maintenant au théorème de non-liquidation lui-même et à sa démonstration. Ce théorème suppose un système de liquidation fonctionnel sur Morpho : les positions sur Morpho sont liquidées si nécessaire. En pratique, cela nécessite un monitoring off-chain fiable et réactif des positions sur Morpho pour que cette hypothèse soit respectée.

Théorème 3.1.2.1 (Non-liquidation)**.** *Si Morpho dispose d'un système de liquidation fonctionnel, alors la position de Morpho sur la pool n'est pas liquidable.*

Démonstration. D'après le lemme d'intégrité pair-à-pair 3.1.2.2, pour chaque token θ, l'équilibre suivant est maintenu :

$$\sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} = \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P}.$$

Les lemmes d'intégrité de la position de la pool 3.1.2.1 fournissent les équations :

$$s_\theta(\overline{M_\psi}) = \sum_u s_\theta^{Pool}(u), \tag{3.5}$$

$$b_\theta(\overline{M_\psi}) = \sum_u b_\theta^{Pool}(u). \tag{3.6}$$

En notant F_θ le facteur de collatéral, le système de liquidation sur Morpho garantit que pour chaque utilisateur u, l'inégalité suivante est satisfaite :

$$\sum_\theta p_\theta \cdot \omega_\theta^B(u) \leq \sum_\theta p_\theta \cdot F_\theta \cdot \omega_\theta^S(u).$$

Par agrégation sur tous les utilisateurs, nous avons :

$$\sum_\theta p_\theta \cdot \sum_u \omega_\theta^B(u) \leq \sum_\theta p_\theta \cdot F_\theta \cdot \sum_u \omega_\theta^S(u).$$

En substituant à partir des équations (3.1), (3.2), puis (3.5) et (3.6), on obtient :

$$\sum_\theta p_\theta \cdot \left(b_\theta(\overline{M_\psi}) \cdot \lambda_\theta^B + \sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} \right) \leq$$
$$\sum_\theta p_\theta \cdot F_\theta \cdot \left(s_\theta(\overline{M_\psi}) \cdot \lambda_\theta^S + \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} \right). \tag{3.7}$$

Puisque pour chaque θ, $F_\theta \leq 1$, il s'ensuit :

$$(1 - F_\theta) \cdot \sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} \geq 0,$$

ce qui implique que :

$$\sum_\theta p_\theta \cdot \left[\sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} - F_\theta \cdot \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{P2P} \right] \geq 0. \tag{3.8}$$

Avec les équations (3.7) et (3.8), nous pouvons conclure que :

$$\sum_\theta p_\theta \cdot b_\theta(\overline{M_\psi}) \cdot \lambda_\theta^B \leq \sum_\theta p_\theta \cdot F_\theta \cdot s_\theta(\overline{M_\psi}) \cdot \lambda_\theta^S,$$

ce qui démontre que la position de Morpho sur la pool n'est pas liquidable. □

3.2 L'algorithme de Mariage

Nous avons précédemment discuté de l'intégrité de l'utilisation de la pool. Nous allons désormais examiner la manière dont Morpho appaire les utilisateurs.

3.2.1 Mise en Pair à Pair des Utilisateurs

L'algoritheoreme de mariage de Morpho est conçu pour appairer les utilisateurs de façon optimale. Les règles de mise en pair sont les suivantes :

- Lors d'un dépôt, Morpho appaire les emprunteurs en attente dans la pool, s'ils existent.
- Lors d'un emprunt, Morpho appaire les prêteurs en attente dans la pool, s'ils existent.
- Lors d'un retrait en pair à pair, deux situations peuvent survenir :
  - S'il existe d'autres prêteurs en attente dans la pool, Morpho les appaire pour prendre la place du prêteur en pair à pair qui se retire.
  - En l'absence d'autres prêteurs en attente, Morpho désappaire les emprunteurs et les reconnecte à la pool.
- Lors d'un remboursement en pair à pair, deux scénarios sont possibles :
  - Si d'autres emprunteurs sont en attente dans la pool, Morpho les appaire pour substituer l'emprunteur en pair à pair.
  - À défaut de prêteurs en attente, Morpho désappaire les prêteurs et les réintègre à la pool.

Il est important de souligner que l'algoritheoreme de mariage ne crée pas de paires d'utili-

sateurs spécifiques, mais se limite à équilibrer le volume total des prêts et des emprunts en pair à pair. De ce fait, les utilisateurs peuvent se retrouver dans un état de pairage partiel.

Exemple 3.2.1.1. *Considérons l'exemple précédent où Bob emprunte uniquement 5 ETH. Alice se retrouve alors partiellement appairée : 5 ETH sont placés dans la pool, tandis que les 5 autres ETH sont en pair à pair.*

3.2.2 Implémentation

L'implémentation de l'algoritheoreme de mariage chez Morpho est relativement simple et suit le schéma suivant. Pour chaque marché, les utilisateurs sont répertoriés dans quatre structures de données différentes : prêteurs sur la pool, prêteurs en pair à pair, emprunteurs sur la pool, et emprunteurs en pair à pair.

Les structures de données employées sont des listes doublement chaînées semi-triées. Cependant, toute autre structure adaptée pourrait être utilisée.

Lorsque l'algoritheoreme veut appairer des utilisateurs à partir de la structure de données `dataStructure` pour un montant donné `montant`, il suit la procédure suivante (en pseudo-code) :

```
function match(dataStructure, montant)
    while (montant > 0 && not isEmpty(dataStructure))
        // Retirer le premier utilisateur de la structure de donnees
        // Deplacer le montant requis d'une structure a l'autre
        montant = montant - pair_a_pair
end
```

Exemple 3.2.2.1. *Supposons que Alice, Bob et Carole aient chacun déposé 10 ETH sur Morpho, ajoutant leur liquidité à la pool.*

La structure de données initiale est :

```
preteurs_sur_la_pool = [(Alice, 10), (Bob, 10), (Carole, 10)]
```

Si David souhaite emprunter 20 ETH, la fonction `match(preteurs_sur_la_pool, 20)` *est invoquée. Alice est d'abord appairée, réduisant le montant restant à 10 ETH. Ensuite, Bob est appairé, ce qui ramène le montant restant à 0 ETH ; l'algoritheoreme s'arrête et la transaction de David est complète, son emprunt étant en pair-à-pair avec Alice et Bob. Carole reste quant à elle sur la pool.*

La structure finale est :

```
preteurs_sur_la_pool = [(Carole, 10)]
```

3.3 Mise à l'échelle du protocole

Nous avons observé que pour chaque ligne de crédit appairée, le protocole doit transférer les fonds d'un utilisateur de la pool vers le pairage. Ce processus de promotion en pair à pair vise à maximiser l'utilisation efficace de la liquidité appairée, à condition qu'il existe un utilisateur correspondant dans l'attente d'être appairé. Ainsi, si n utilisateurs ont déposé un total de 1000 DAI dans la pool et qu'un emprunt de 1000 DAI est demandé, il faudra ajuster les lignes de crédit de ces n utilisateurs, avec une complexité algoritheoremique en $O(n)$.

3.3.1 Limitation du Pairage

La complexité linéaire représente un problème d'équité pour le protocole. Cela signifie que le coût d'utilisation de Morpho dépend du nombre d'utilisateurs à appairer. Une grande diversité d'utilisateurs peut entraîner un coût très variable, pouvant doubler ou tripler.

Dans le pire des cas, on pourrait atteindre la limite maximale de calcul que l'Ethereum Virtual Machine (EVM) peut gérer dans un seul bloc.

Sur Ethereum, cela représente 30M de gas, nécessitant environ 2000 utilisateurs à appairer pour parvenir à cette limite. Dans de telles circonstances, l'utilisation de Morpho deviendrait impraticable. Plus problématique, cela signifierait que les utilisateurs ne pourraient pas retirer leurs fonds, remettant en question l'amélioration de Pareto promise par le protocole. En conséquence, cela engendrerait deux problèmes majeurs :

— Variabilité importante du coût d'utilisation du protocole.
— Le protocole cesserait d'être une amélioration de Pareto en pratique.

De plus, cela constituerait une vulnérabilité pour le protocole. En effet, il suffirait de positionner des montants minimaux sur Ethereum (1e-18 ETH) et de créer de nombreux utilisateurs dans la structure de données. Ce type d'attaque, connu sous le nom d'attaque de Griefing, n'apporte aucun bénéfice direct à l'attaquant mais peut causer un préjudice significatif au produit concerné, dans ce cas Morpho.

3.3.2 Définition d'une limite de pairage

Une question préliminaire pourrait être : pourquoi chercher à maximiser le pairage ? La raison principale est d'optimiser les taux d'intérêt au maximum. Toutefois, atteindre le pairage maximal n'est pas une condition sine qua non pour le protocole. Imaginons que nous décidions de limiter le pairage à 10 utilisateurs par transaction : dans ce cas, Morpho conserverait toutes ses caractéristiques essentielles. Si plus de 10 utilisateurs s'avéraient nécessaires, alors l'efficacité de l'algoritheoreme de pairage ne serait pas pleine et entière.

Lors du pairage suivant, il serait alors possible d'optimiser davantage l'efficacité.

Définition 3.3.2.1. *Soit $N \in \mathbb{N}$ le nombre maximal d'utilisateurs à pairer dans une transaction unique par l'algoritheoreme de pairage.*

Le processus de pairage serait alors ajusté comme suit :

```
match(dataStructure, montant):
    i = 0
    while(montant > 0 && not dataStructure.empty() && i < N):
        // Retire le premier utilisateur de la structure de donnees
        // Transfere le montant requis d'une structure de donnees vers l'autre
        montant = montant - montant_pair_a_pair
        i += 1
```

Efficacité de Morpho

Lorsque la valeur de N est atteinte, on considère que l'algoritheoreme de pairage n'a pas atteint une efficacité maximale.

Dans un marché où le pairage est possible, il est crucial de disposer d'une métrique d'efficacité.

Morpho peut être considéré comme optimal si 100% des prêteurs **ou** 100% des emprunteurs sont en pairage direct. Cela signifie que l'excédent de liquidités reste dans la pool sans pouvoir être apparié immédiatement. Si l'algoritheoreme de pairage pouvait fonctionner indéfiniment, l'efficacité serait toujours de 100%, car tous les pairages possibles seraient réalisés.

L'efficacité est définie comme le rapport entre la liquidité en pairage direct et la liquidité potentiellement appairable.

La formule de l'efficacité, où S_{p2p} représente la liquidité des prêteurs appariés, S_{pool} celle des prêteurs dans la pool, B_{p2p} celle des emprunteurs appariés, et B_{pool} celle des emprunteurs dans la pool, est la suivante :

$$e = \frac{\min(S_{p2p}, B_{p2p})}{\min(S_{p2p} + S_{pool}, B_{p2p} + B_{pool})} \tag{3.9}$$

Dans le cas où $N = \infty$, on a $S_{p2p} = B_{p2p}$ et $\min(B_{pool}, S_{pool}) = 0$, ce qui nous donne bien $e = 1$.

Le cas particulier du désappariement

Le processus de désappariement est légèrement plus complexe. Comme nous l'avons vu dans la section 3.3.2, lorsqu'un utilisateur souhaite retirer ses fonds et que la limite N est atteinte, l'algoritheoreme de pairage ne peut plus continuer et l'utilisateur ne peut pas récupérer l'intégralité de ses fonds immédiatement, car la liquidité reste dans la pool.

Considérons l'exemple suivant pour illustrer cette situation :

Exemple 3.3.2.1. *Supposons la structure de pairage des prêteurs d'ETH suivante :*

```
preteurs_pair_a_pair = [(Alice, 10), (Bob, 10), (Carole, 5)]
```

Avec $N = 2$. Imaginons que Charlie souhaite retirer 25 ETH. Il peut récupérer 10 ETH auprès d'Alice et 10 ETH auprès de Bob, mais ensuite, l'algoritheoreme de pairage s'arrête. Morpho doit alors emprunter 20 ETH de la pool pour désapparier Alice et Bob. Par conséquent, Charlie ne peut pas retirer les 5 ETH restants, car ils manquent dans la transaction.

Impact sur la liquidation La liquidation sur Morpho a pour objectif de rembourser la dette d'un utilisateur en utilisant une partie de son dépôt. Lorsque la dette ou le dépôt est en pair-à-pair, la complexité du processus de remboursement augmente, ce qui pourrait empêcher la liquidation de certains utilisateurs, compromettant ainsi le théorème de non liquidation.

Exemple 3.3.2.2. *Considérons la structure des emprunteurs en pair-à-pair d'ETH suivante :*

```
preteurs_pair_a_pair = [("Alice", 10), ("Bob", 10), ("Carole", 5)]
```

Supposons que $N = 2$. Imaginons Charlie, un utilisateur liquidable avec une dette de 25 ETH en pair-à-pair. Un liquidateur veut rembourser la dette de Charlie, mais l'algoritheoreme limite le remboursement à 20 ETH depuis la pool de liquidité. Par conséquent, Charlie ne peut être liquidé qu'à hauteur de 20 ETH, diminuant ainsi le profit potentiel du liquidateur.

Dans l'exemple ci-dessus, on pourrait envisager une liquidation partielle suivie d'une liquidation en cascade. Cependant, si la liquidation partielle n'est pas rentable pour le liquidateur, alors l'emprunteur risque de ne jamais être liquidé, mettant en péril la fonctionnalité essentielle du système de liquidation de Morpho.

Conclusion

En conclusion, cette partie a mis en exergue les contraintes intrinsèques du protocole Morpho engendrées par la limite de pairage définie par N. L'instauration d'une telle limite, bien que justifiable pour des raisons de performances et de gestion transactionnelle au sein du réseau, a révélé des enjeux considérables. Ces derniers concernent principalement l'efficacité du pairage et la capacité de liquidation, mettant en lumière la nécessité d'une solution évolutive. Il apparaît dès lors crucial que le protocole puisse s'adapter et répondre de manière efficace aux exigences dynamiques du marché.

Le défi qui se profile consiste à élaborer une solution capable non seulement d'accompagner l'expansion naturelle de l'écosystème Morpho, mais aussi de préserver, voire d'améliorer, l'efficacité du protocole dans toutes les conditions de marché. La section suivante de ce mémoire se concentrera sur cette problématique de mise à l'échelle. Nous examinerons une stratégie susceptible de contourner les limitations actuelles sans compromettre la sécurité ou la fiabilité du système. À travers l'exploration d'une approche innovante, nous chercherons à gérer efficacement les volumes importants de pairages et de liquidations potentielles, en gardant une perspective globale sur leur intégration harmonieuse au sein de l'infrastructure existante de Morpho.

En quête de solutions pour les problèmes de mise à l'échelle, nous nous avancerons dans un territoire où technologie et inventivité se rencontrent pour forger de nouvelles voies de croissance et de robustesse pour le protocole, assurant ainsi sa pérennité dans l'univers compétitif de la finance décentralisée.

CHAPITRE 4 MISE À L'ÉCHELLE DE L'ALGORITHME DE MARIAGE

Nous allons décrire ici le fonctionnement du méchanisme Delta utilisé en méthode de repli en cas de durée excessive d'une transaction au sein de l'algorithme de mariage. Le problème a été posé en septembre 2021, et est le sujet principale de ce Mémoire.

4.1 Le méchanisme "Delta"

Le pairage ou le déparaillage des utilisateurs pair à pair n'est pas une opération à temps constant, et les transactions importantes peuvent ne pas trouver suffisamment de liquidité compte tenu des ressources allouées (en gas sur Ethereum). Le mécanisme delta est conçu pour résoudre ce problème.

La justification d'une limite sur le calcul peut se faire dans différents contextes : cela peut être fait pour économiser du temps CPU, pour limiter la puissance totale utilisée ou pour prouver qu'une implémentation particulière se termine en ajoutant d'abord une borne maximale N.

En utilisant cette limite, nous pouvons alors décider à tout moment si l'algorithme doit s'arrêter. Sur Morpho, nous appliquons le mécanisme delta pour limiter l'utilisation de gas dans l'EVM [23].

Cette section donne d'abord une description de haut niveau du problème et du mécanisme delta, ce qui nous permet d'en déduire l'impact sur Morpho, ses théorèmes et ses taux pair à pair.

Nous passons ensuite à une description plus détaillée de la manière dont cela est mis en œuvre, et nous concluons en mettant à jour Morpho pour prendre en compte le mécanisme delta, en introduisant les variables nécessaires, en redémontrant les théorèmes et en modifiant l'algorithme de Mariage.

4.1.1 Description

Trouver un pairage peut être un calcul intensif, et nous expliquons ici comment nous pouvons arrêter l'algorithme de mariage plus tôt pour atténuer son coût.

Cela revient à trouver la liquidité manquante lorsque l'algorithme n'a pas renvoyé un montant suffisamment élevé.

Lors du prêt ou de l'emprunt, si des liquidités insuffisantes ont pu être trouvées, nous avons déjà la pool de secours : Morpho remplit le reste du dépot non pairé avec un prêt ou un

emprunt sur la pool, et les balances des utilisateurs sont mis à jour en conséquence.

Pour le retrait et le remboursement, puisque ces fonctions se terminent par une partie correspondante censée trouver la liquidité demandée, nous devons trouver une autre solution de repli.

En effet, nous aurions besoin de modifier l'équilibre entre les utilisateurs non pairés et les états des structures de données, mais nous ne pouvons pas à cause de la contrainte de calcul.

L'idée est de poursuivre l'étape de dépareillage en augmentant la position de Morpho sur la pool en conséquence.

Cela signifie que le taux subi par les prêteurs devrait diminuer en cas de "dépareillage-remboursement", et le taux subi par les emprunteurs devrait augmenter en cas de "dépareillage-retrait". Le mécanisme delta est conçu pour tenir compte de ce changement.

Pour résumer, le mécanisme delta pair à pair fonctionne comme suit :

nous opérons le repli sur la pool pour remplir la demande de liquidité, même si nous ne pouvons pas dissocier suffisamment d'utilisateurs pair à pair. Il introduit une différence [1] entre la liquidité pair à pair évoluant aux taux de Morpho et la somme des balances pair à pair des utilisateurs de ce côté du marchés. Afin d'y remédier, les tarifs vécus par tous les utilisateurs pair à pair sont adaptés (le taux global pair à pair est réduit).

Concrètement, considérons le cas où un utilisateur se retire (resp. rembourse), et sa liquidité est pairée en pair à pair. Supposons également qu'il n'y ait pas suffisamment de liquidités des prêteurs (resp. emprunteurs) sur la pool pour le remplacer. Ainsi, les lignes de crédit pair à pair seront rompues, et les emprunteurs pair à pair (resp. fournisseurs) seront reconnectés à la pool. Si l'algorithme de mariage ne trouve pas suffisamment de liquidités à cette étape, alors Morpho emprunte (resp. retire) toutes les liquidités demandées restantes sur la pool. Cela signifie que tous les emprunteurs pair à pair (resp. prêteurs) ont maintenant une partie de leur emprunt (resp. dépot) sur la pool et obtiennent désormais un taux moins avantageux, car une partie est sur la pool.

Regardons comment cela se traduit sur la comptabilité des positions.

1. Le nom “delta” fait référence à cette différence.

Soit A l'ensemble des utilisateurs de Morpho, et soient S^{P2P} et S^{Pool} l'offre totale d'utilisateurs Morpho en pair à pair et sur la pool :

$$S^{P2P} = \sum_{a \in A} s^{P2P}(a)$$
$$S^{Pool} = \sum_{a \in A} s^{Pool}(a)$$

De même, supposons que B^{P2P} et B^{Pool} soient le total des emprunts :

$$B^{P2P} = \sum_{a \in A} b^{P2P}(a)$$
$$B^{Pool} = \sum_{a \in A} b^{Pool}(a)$$

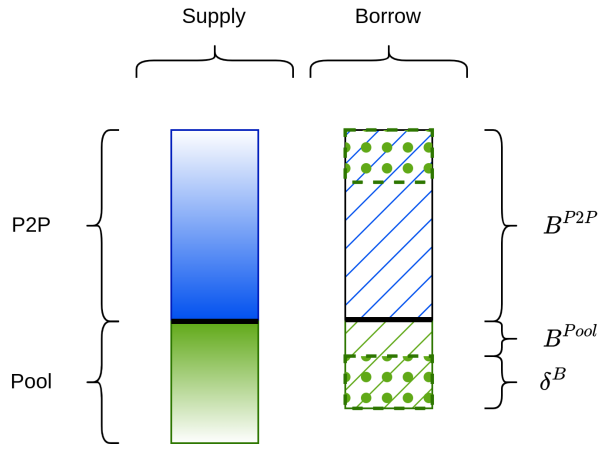
Posons également $s(\overline{M_\psi})$ la position de Morpho en dépôt sur la pool, et $b(\overline{M_\psi})$ sa position en emprunt.

Rappelez-vous que $s(a)$ et $b(a)$ représentent les balances de prêt et d'emprunt de l'utilisateur a sur la pool. Ces grandeurs sont définies indépendamment de Morpho.

Le delta de prêt pair à pair δ^S et le delta d'emprunt pair à pair δ^B sont définis comme suit :

$$S^{Pool} + \delta^S = s(\overline{M_\psi})$$
$$B^{Pool} + \delta^B = b(\overline{M_\psi})$$

Ces notations sont représentées sur le graphique suivant, montrant uniquement le cas du delta d'emprunt pour simplifier.



Exemple 4.1.1.1. *Prenons un exemple simple. Supposons que l'algorithme de mariage puisse*

dissocier au plus trois utilisateurs ($N = 3$). Les trois situations successives suivantes nous donnent un exemple du fonctionnement du mécanisme delta.

Prêteurs	sur la pool	en p2p
utilisateur 1	0	60

Position de Morpho sur la pool : 0

Emprunteurs	sur la pool	en p2p
utilisateur 2	0	10
utilisateur 3	0	10
utilisateur 4	0	10
utilisateur 5	0	10
utilisateur 6	0	10
utilisateur 7	0	10

Position de Morpho sur la pool : 0

TABLEAU 4.1 Algorithme de Mariage : Situation initiale

Prêteurs	sur la pool	en p2p
utilisateur 1	0	30

Position de Morpho sur la pool : 0

Emprunteurs	sur la pool	en p2p
utilisateur 2	10	0
utilisateur 3	10	0
utilisateur 4	10	0
utilisateur 5	0	10
utilisateur 6	0	10
utilisateur 7	0	10

Position de Morpho sur la pool : 30

TABLEAU 4.2 Après un retrait via l'algorithme de mariage (1 utilisateur retire 60)

Prêteurs	sur la pool	en p2p
utilisateur 1	0	0

Position de Morpho sur la pool : 0

Emprunteurs	sur la pool	en p2p
utilisateur 2	10	0
utilisateur 3	10	0
utilisateur 4	10	0
utilisateur 5	0	10
utilisateur 6	0	10
utilisateur 7	0	10

Position de Morpho sur la pool : 60

TABLEAU 4.3 Après le retrait total

Après l'ensemble de la transaction, le delta d'emprunt pair à pair est égal à 30, et tout le montant pair à pair est sur la pool.

Il existe trois situations dans lesquelles Morpho traite le mécanisme delta.

Augmentation du Delta Après un passage dans l'algorithme de Mariage lors d'un remboursement ou d'un retrait, s'il n'y a pas assez d'utilisateurs pair à pair qui pourraient être pairé, nous augmentons le delta de la liquidité manquante.

Réalisation du Delta Lorsque de nouvelles liquidités arrivent (lors d'un prêt ou d'un emprunt), le delta de l'autre côté du marché est "pairé", ce qui le réduit.

Réduction du Delta Lorsque certaines liquidités pair à pair sortent, le delta correspondant est réduit. Par exemple, le retrait de dépots pair à pair réduira d'abord le delta du coté du prêt sur la pool.

4.2 Implémentation sur Morpho

Pour s'assurer que les coûts de gas des transactions sont limités, nous nous assurons que le gas utilisé pour l'algorithme de mariage ne dépasse pas une limite prédéfinie N. Si le nombre N d'itérations est atteint, alors l'algorithme de mariage va renvoyer un montant correspondant inférieur à la liquidité demandée. Étant donné que d'autres parties du code ont également un coût de gas borné, nous savons que les coûts de gas globaux des opérations de Morpho sont bornés. De cette façon, Morpho peut évoluer à mesure que de plus en plus d'utilisateurs entrent sur les différents marchés et garantir sa mise à l'échelle, sans modification du coût d'utilisation.

La mise en pair à pair n'est pas obligatoire pour que les opérations réussissent. Ainsi, nous introduisons la possibilité pour les utilisateurs de Morpho de choisir le paramètre n, c'est-à-dire le nombre d'itérations qu'ils sont prêts à payer pour la mise en pair à pair en appelant les fonctions $\texttt{supply}^M$ et $\texttt{borrow}^M$.

Cela ne peut pas être utilisé dans $\texttt{withdraw}^M$, $\texttt{repay}^M$ et $\texttt{liquidate}^M$. En effet, pour ces fonctions, il n'y a pas d'incitation économique à pairer ou dépareiller plus d'utilisateurs (en fait, il y a une incitation à ne pas le faire à cause du coût du gas). Ainsi, pour ces fonctions, le nombre maximum d'itération N est fixé. Il peut être 0 pour $\texttt{liquidate}^M$ afin de réduire le coût du gas de la liquidation.

Les implémentations des fonctions de mise en pair à pair doivent être modifiées pour tenir compte le nombre maximum d'itération afin qu'elles cessent de s'exécuter avant d'avoir itérer à travers toute la structure de donnée.

Nous devons également suivre les deltas dans les nouvelles variables de stockage :

$$\delta^S \in \Theta_\psi \to \mathbb{R}_+$$
$$\delta^B \in \Theta_\psi \to \mathbb{R}_+$$

pour respectivement le delta de prêt pair à pair et le delta d'emprunt pair à pair. Leurs valeurs sont exprimées en unités à l'échelle de la pool, car elles évoluent aux taux de la pool . De plus, pour calculer la part du delta et mettre à jour les indices, nous devons suivre les montants totaux mis en pair à pair. Pour ce faire, nous introduisons deux nouvelles variables dans le stockage de Morpho :

$$S^{P2P} \in \Theta_\psi \to \mathbb{R}_+$$

pour le montant total de l'offre pair à pair et

$$B^{P2P} \in \Theta_\psi \to \mathbb{R}_+$$

pour le montant total d'emprunt pair à pair.

ces montants représentent la reduction de toutes les balances utilisateurs en pair à pair qui sont stockés dans les structures de données.

Enfin, nous définissons l'indice d'offre pair à pair qui prend en compte le mécanisme delta (en plus du curseur pair à pair et du facteur de réserve) :

$$\lambda^{\gamma^S}$$

De même, nous avons l'indice d'emprunt pair à pair prenant en compte le mécanisme delta :

$$\lambda^{\gamma^B}$$

La sous-section 4.3 détaille comment ces indices pair à pair sont mis à jour.

Réécrivons les fonctions principales de Morpho avec l'implémentation du mécanisme delta.

4.2.1 Prêt (Supply)

Description

1. **Prêt pair à pair :**
   (a) **Pairer le delta d'emprunt pair à pair :** Morpho paire l'utilisateur avec le delta d'emprunt pair à pair et le réduit. Ceci est fait en premier pour maintenir le delta aussi bas que possible. Un `remboursement` sur la pool est effectué pour réduire le surplus d'emprunt sur la pool induit par le delta d'emprunt pair à pair, en utilisant la liquidité de l'utilisateur qui prête. Notez que cela permet aux utilisateurs réactifs d'ignorer la file d'attente des utilisateurs sur la pool et d'être instantanément mis en pair à pair.
   (b) **Pairer des emprunteurs sur la pool :** Morpho paire la liquidité entrante avec une dette sur la pool. L'algorithme de mariage essaie de faire correspondre autant de dettes que possible aux emprunteurs de la pool. Un `remboursement` sur la pool est effectué pour en déconnecter les emprunteurs pairés, en utilisant la liquidité du prêteur.
2. **Prêt sur la pool :**
   (a) **Pas de pairage :** La liquidité restante est déposée sur la pool sous-jacente. Il n'y a pas de mise en pair à pair ici.

```
pret^M_θ(utilisateur, montant):
    mettre_a_jour_λ^M_θ()
    transferer_θ(utilisateur, M_ψ, montant)

    // pret pair a pair

    //// Pairer le delta d'emprunt pair a pair
    pair_a_pair = min(δ^B_θ * λ^B_θ, montant)
    δ^B_θ -= pair_a_pair / λ^B_θ
    s^P2P_θ(utilisateur) += pair_a_pair / λ^γS_θ
    S^P2P += pair_a_pair / λ^γS_θ
    montant -= pair_a_pair
    rembourser_θ(M_ψ, pair_a_pair)

    //// Pairer des emprunteurs sur la pool
    pair_a_pair = algorithme_mariage_emprunteurs_θ(montant, N)
```

s_θ^{P2P}(utilisateur) += pair_a_pair / $\lambda_\theta^{\gamma_S}$
S^{P2P} += pair_a_pair / $\lambda_\theta^{\gamma_S}$
B^{P2P} += pair_a_pair / $\lambda_\theta^{\gamma_B}$
montant -= pair_a_pair
rembourser$_\theta$($\overline{M_\psi}$, pair_a_pair)

// Pret sur la pool

s_θ^{Pool}(utilisateur) += montant / λ_θ^{S}
pret$_\theta$($\overline{M_\psi}$, montant)

4.2.2 Emprunt (Borrow)

Description

1. **Emprunt pair à pair :**
   (a) **Pairer le delta de prêt pair à pair :** Morpho "paire" l'utilisateur avec le delta de prêt pair à pair et le réduit. Ceci est fait en premier pour maintenir le delta aussi bas que possible. Un `prélèvement` sur la pool est effectué pour réduire le surplus de prêt sur la pool induit par le delta de prêt pair à pair. Notez que cela permet aux utilisateurs réactifs d'ignorer la file d'attente des utilisateurs sur la pool et d'être instantanément mis en pair à pair.
   (b) **Pairer des prêteurs sur la pool :** Morpho paire à la demande entrante avec une certaine offre sur la pool. L'algorithme de mariage essaie d'apparier autant de liquidités que possible des prêteurs de la pool. Un `retrait` sur la pool est effectué pour en déconnecter les fournisseurs pairés à la fin de l'algorithme.
2. **Emprunt sur la pool :**
   (a) **Pas de pairage :** La demande restante est retirée sur la pool sous-jacente avec un appel à $emprunt_\theta$.

Description formelle

```
emprunter_θ^M(utilisateur, montant):
    mettre_a_jour_λ_θ^M()
    if montant * prix_θ > capacite_emprunt^M(a):
        return
    montant_initial = montant

    // Emprunt pair a pair

    //// Pairer le delta de pret pair a pair :
    pair_a_pair = min(δ_θ^S * λ_θ^S, montant)
    δ_θ^S -= pair_a_pair / λ_θ^S
    b_θ^P2P(utilisateur) += pair_a_pair / λ_θ^γB
    B^P2P += pair_a_pair / λ_θ^γB
    montant -= pair_a_pair
    retirer_θ(M_ψ, pair_a_pair)

    //// Pairer des preteurs sur la pool
    pair_a_pair = algorithme_mariage_preteurs_θ(montant, N)
    b_θ^P2P(utilisateur) += pair_a_pair / λ_θ^γB
    S^P2P += pair_a_pair / λ_θ^γS
    B^P2P += pair_a_pair / λ_θ^γB
    montant -= pair_a_pair
    retirer_θ(M_ψ, pair_a_pair)

    // Emprunt sur la pool

    b_θ^Pool(utilisateur) += montant / λ_θ^B
    emprunter_θ(M_ψ, montant)

    transferer_θ(M_ψ, utilisateur, montant_initial)
```

4.2.3 Retirer (Withdraw)

Description

1. **Retirer sur la pool :**
   (a) **Pas de match :** Si une partie de la liquidité de l'utilisateur est fournie sur la pool,

Morpho retire la partie correspondante de sa position sur la pool sous-jacente. Il privilégie cette option pour maximiser la liquidité pair à pair de l'utilisateur et, par conséquent, l'efficacité de son capital. Un `retrait` sur la pool est effectué.

(b) **Réduire le delta de prêt pair à pair :** Morpho réduit le delta de prêt pair à pair s'il y en a un. Il privilégie cette option avant le retrait du pair à pair pour minimiser le delta. Un `retrait` sur la pool est effectué pour réduire le delta de prêt pair à pair.

2. **Retrait pair à pair :**

(a) **Transmettre le pairage :**

i. **Promotion de prêteurs :**Si l'utilisateur est déjà en pair à pair, Morpho le remplace par d'autres prêteurs sur la pool. Il privilégie cette option avant la réduction du pairage pour maximiser la liquidité totale en pair à pair. L'algorithme de mariage essaie de pairer autant de liquidités que possible des fournisseurs sur la pool, dans la limite N. La liquidité est récupérée par un appel à `retirer` sur la pool sous-jacente.

(b) **Reduction du pairage :**

i. **Dépareiller les emprunteurs :** Morpho rompt les lignes de crédit pair à pair de l'utilisateur qui se retire avec les emprunteurs pairés et les reconnecte à la pool. L'algorithme de mariage essaie de dissocier autant de dettes que possible des emprunteurs pair à pair. Un `emprunt` sur la pool est effectué pour reconnecter les emprunteurs à la pool. Le montant emprunté est transféré à l'utilisateur.

ii. **Augmenter le delta d'emprunt pair à pair :** Supposons que le nombre d'itérations disponibles pour l'algorithme de mariage n'était pas suffisant pour libérer suffisamment de liquidités. Dans ce cas, Morpho effectue toujours le retrait en reconnectant les emprunteurs pair à pair à la pool via un emprunt sur la pool, et augmente le delta d'emprunt pair à pair. Une partie des emprunts pairés en pair à pair est maintenant sur la pool, et les indices seront mis à jour en conséquence à partir de maintenant, augmentant à un rythme plus élevé si on considère que le taux d'emprunt pair à pair est plus faible que le taux d'emprunt sur la pool.

Description formelle

$\texttt{retrait}_{\theta}^{M}$`(utilisateur, montant):`

```
mettre_a_jour_λ_θ^M()
if montant * facteur_de_collateral(\theta) * prix_θ > capacite_retrait^M(utilisateur):
    return
montant_initial = montant

// Retirer sur la pool

retrait_sur_la_pool = min(montant, s_θ^Pool(utilisateur) * λ_θ^S)
s_θ^Pool(utilisateur) -= retrait_sur_la_pool / λ_θ^S
montant -= retrait_sur_la_pool
retrait_θ(M_ψ, retrait_sur_la_pool)

// Reduire le delta de pret pair a pair
pair_a_pair = min(δ_θ^S * λ_θ^S, montant)
δ_θ^S -= pair_a_pair / λ_θ^S
s_θ^P2P(utilisateur) -= pair_a_pair / λ_θ^γS
S^P2P -= pair_a_pair / λ_θ^γS
montant -= pair_a_pair
retrait_θ(M_ψ, pair_a_pair)

// Retrait pair à pair

//// Transmettre le pairage

// Promotion de preteurs
pair_a_pair = algorithme_mariage_preteurs_θ(montant, N)
s_θ^P2P(utilisateur) -= pair_a_pair / λ_θ^γS
montant -= pair_a_pair
retrait_θ(M_ψ, pair_a_pair)

//// Retrait du pairage

// Depareillement des emprunteurs
depairee = algorithme_mariage_emprunteurs_dissociation_θ(montant, N)
// Augmentation du delta d'emprunt
if montant - depairee > 0:
```

δ_θ^B += (montant - depairee) / λ_θ^B
S^{P2P} -= montant / $\lambda_\theta^{\gamma_S}$
B^{P2P} -= depairee / $\lambda_\theta^{\gamma_B}$
s_θ^{P2P}(utilisateur) -= montant / $\lambda_\theta^{\gamma_S}$
emprunt$_\theta$($\overline{M_\psi}$, montant)

transferer$_\theta$($\overline{M_\psi}$, utilisateur, montant_initial)

4.2.4 Remboursement (Repay)

Description

1. **Rembourser sur la pool :**
   (a) **Pas de match :** Si une partie de la dette de l'utilisateur est empruntée sur la pool, Morpho rembourse les liquidités sur la pool sous-jacente. Il privilégie cette option pour maximiser la liquidité et l'efficacité du capital de l'utilisateur. Un `remboursement` est effectué sur la pool.
   (b) **Réduire le delta d'emprunt pair à pair** Morpho réduit le delta d'emprunt pair à pair s'il y en a un. Il privilégie cette option avant le remboursement du transfert pour minimiser le delta. Un `remboursement` sur le pool est effectué pour réduire le delta d'emprunt pair à pair.
2. **Paiement des frais :** Les frais facturés par Morpho sont comptabilisés lorsque les utilisateurs remboursent leurs positions pair à pair. Le montant total des frais accumulés sur tous les utilisateurs peut être calculé en soustrayant le montant réel (en soustrayant le delta) de l'offre en pair à pair au montant réel des emprunts en pair à pair. Cette différence provient du facteur de réserve qui fait croître l'indice d'emprunt pair à pair à une vitesse plus élevée que l'indice d'offre pair à pair. La position pair à pair de l'utilisateur est mise à jour en fonction des frais remboursés, ce qui rend cette opération transparente lors de l'interaction avec le protocole.
3. **Remboursement pair à pair :**
   (a) **Transmettre le pairage :**
      i. **Promotion des emprunteurs :** Si l'utilisateur est pairé en pair à pair, Morpho remplace les lignes de crédit pair à pair par d'autres emprunteurs de la pool. Il privilégie cette option avant le breaking repay afin de maximiser la liquidité pair à pair totale. L'algorithme de mariage essaie de faire correspondre

autant de dettes que possible aux emprunteurs de la pool. Un `repay` sur la pool est effectué pour réduire la dette de ces emprunteurs sur la pool.

(b) **Reduction du pairage :**

i. **Dépareiller les prêteurs :** Morpho rompt les lignes de crédit pair à pair de l'utilisateur qui rembourse avec les fournisseurs pairés restants et dépose la liquidité inégalée sur la pool sous-jacente. L'algorithme de mariage essaie de dissocier autant de liquidités que possible des fournisseurs pair à pair. Morpho `supply` sur la pool, pour reconnecter les fournisseurs à la pool.

ii. **Augmentation du delta des prêteurs :** Si le nombre d'itération disponible pour le pairage n'était pas suffisant pour libérer suffisamment de liquidités, Morpho procède tout de même à un dépot sur la pool (reconnexion des fournisseurs pair à pair sur la pool) et augmente le delta de prêt pair à pair. Une partie de l'offre pairée est maintenant sur la pool, et les indices seront mis à jour en conséquence à partir de maintenant, augmentant à un rythme plus lent.

Description formelle

```
remboursement_θ^M(utilisateur, montant):
    mettre_a_jour_λ_θ^M()
    if montant > b_θ^P2P(utilisateur) * λ^γB + b^Pool(utilisateur) * λ^B:
        return
    transferer_θ(utilisateur, M_ψ, montant)

    // Rembourser sur la pool

    dette_utilisateur_pool = min(montant, b_θ^Pool(utilisateur) * λ_θ^B)
    b_θ^Pool(utilisateur) -= dette_utilisateur_pool / λ_θ^B
    montant -= dette_utilisateur_pool
    rembourser_θ(M_ψ, dette_utilisateur_pool)

    // Reduire le delta d'emprunt pair a pair
    pair_a_pair = min(δ_θ^B * λ_θ^B, montant)
    δ_θ^B -= pair_a_pair / λ_θ^B
    b_θ^P2P(utilisateur) -= pair_a_pair / λ_θ^γB
    B^P2P -= pair_a_pair / λ_θ^γB
    montant -= pair_a_pair
```

```
rembourser_θ(M_ψ, pair_a_pair)

// Paiement des frais
frais = (B^P2P * λ_θ^γB − δ_θ^B * λ_θ^B) − (S^P2P * λ_θ^γS − δ_θ^S * λ_θ^S)
frais_rembourses = min(frais, montant)
b_θ^P2P(utilisateur) -= frais_rembourses / λ_θ^γB
B^P2P -= frais_rembourses / λ_θ^γB
montant -= frais_rembourses

// Remboursement pair a pair

//// Transmettre le pairage

// Promotion des emprunteurs
pair_a_pair = algorithme_mariage_emprunteurs_θ(montant, N)
b_θ^P2P(utilisateur) -= pair_a_pair / λ_θ^γB
montant -= pair_a_pair
rembourser_θ(M_ψ, pair_a_pair)

//// Reduction du pairage

// Depareiller les preteurs
depairee = algorithme_mariage_preteurs_dissociation_θ(montant, N)
// Augmentatio du delta d'emprunt
if montant - depairee > 0:
    δ_θ^S += (montant - depairee) / λ_θ^S
S^P2P -= depairee / λ_θ^γS
B^P2P -= montant / λ_θ^γB
b_θ^P2P(utilisateur) -= montant / λ_θ^γB
supply_θ(M_ψ, montant)
```

4.3 Impact sur les taux

Nous avons vu lors de la description du mechanisme que le delta introduisait une différence entre la comptabilité faite au sein de Morpho des utilisateurs sur la pool / en pair à pair, et ce qu'il est réellement de la position de Morpho sur la Pool. Cela veut dire que l'emprunt

(resp. la dette) de Morpho sur la pool n'est pas le reflet exacte de la somme des utilisateurs. En effet, on a des intérêts pair à pair qui vont forcément être correlé à la pool lorsqu'un delta est présent. Nous allons voir dans cette partie comment cela se reflette sur les taux, et les indices pair à pair.

Supposons que λ^{γ^S} et λ^{γ^B} soient l'indice de prêt pair à pair et l'indice d'emprunt pair à pair. Cette notation est introduite pour tenir compte du mécanisme delta (en plus du reserve factor et du curseur p2p). Nous voulons calculer ces indices. Nous définissons la part d'offre et d'emprunt du delta comme suit :

$$\gamma^S = \frac{\delta^S \cdot \lambda^S}{S^{P2P} \cdot \lambda^{\gamma^S}}$$
$$\gamma^B = \frac{\delta^B \cdot \lambda^B}{B^{P2P} \cdot \lambda^{\gamma^B}}$$

Nous voulons que la partie de la liquidité pairée correspondant au delta croît à une vitesse qui dépend du taux de la pool (r^S et r^B). Le reste devrait croître à une vitesse qui dépend du taux pair à pair (r^{ρ^S} et r^{ρ^B}). Cela implique que les taux résultants sont, pour les indices de prêt :

$$\begin{aligned} r^{\gamma^S} &= (1-\gamma^S) r^{\rho^S} + \gamma^S r^S \\ &= (1-\gamma^S)((1-\alpha+\rho\alpha) r^S + (\alpha - \rho\alpha) r^B) + \rho r^S \\ &= (1-(1-\gamma^S)(1-\rho)\alpha) r^S + (1-\gamma^S)(1-\rho)\alpha r^B \end{aligned}$$

et pour les index d'emprunt :

$$\begin{aligned} r^{\gamma^B} &= (1-\gamma^B) r^{\rho^B} + \gamma^B r^B \\ &= (1-\gamma^B)((1-\rho)(1-\alpha) r^S + (\rho + \alpha - \rho\alpha) r^B) + \gamma^B r^B \\ &= (1-\gamma^B)(1-\rho)(1-\alpha) r^S + (1-(1-\gamma^B)(1-\rho)(1-\alpha)) r^B \end{aligned}$$

Prenons $t, t' \in \mathbb{N}$ avec $t' < t$. En utilisant le théorème 3.0.3.1, on obtient :

$$\lambda_t^{\gamma^S} \approx \lambda_{t'}^{\gamma^S} [(1-(1-\gamma_{t'}^S)(1-\rho)\alpha) \frac{\lambda_t^S}{\lambda_{t'}^S} + (1-\gamma_{t'}^S)(1-\rho)\alpha \frac{\lambda_t^B}{\lambda_{t'}^B}]$$
$$\lambda_t^{\gamma^B} \approx \lambda_{t'}^{\gamma^B} [(1-\gamma_{t'}^B)(1-\rho)(1-\alpha) \frac{\lambda_t^S}{\lambda_{t'}^S} + (1-(1-\gamma_{t'}^B)(1-\rho)(1-\alpha)) \frac{\lambda_t^B}{\lambda_{t'}^B}]$$

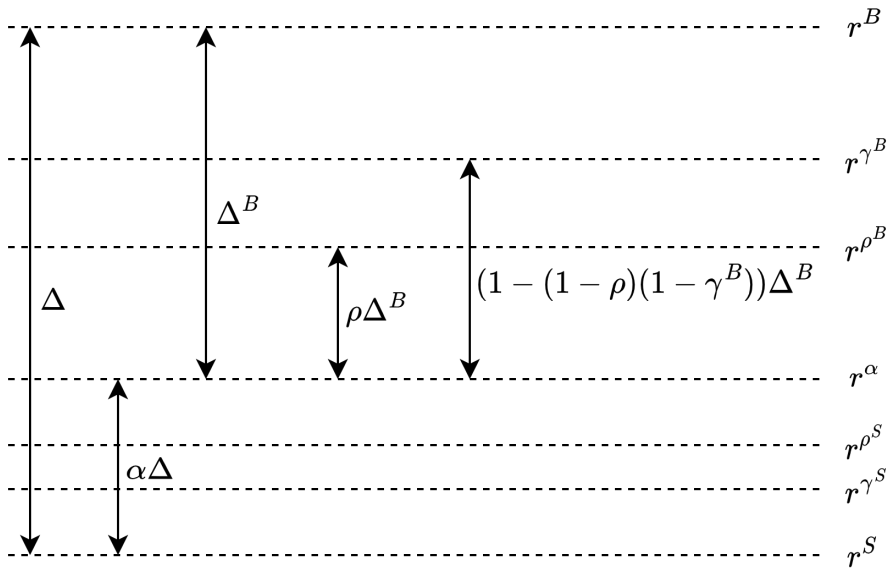
Ces formules sont des calculs efficaces qui proviennent de l'approximation des taux souhaités

en tenant compte du facteur de réserve et du mécanisme delta. Le protocole Morpho ne met pas à jour les taux mais met à jour les index à la place, en utilisant les formules ci-dessus.

$$\lambda_t^{\gamma^S} = \lambda_{t'}^{\gamma^S}[(1-(1-\gamma_{t'}^S)(1-\rho)\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + (1-\gamma_{t'}^S)(1-\rho)\alpha\frac{\lambda_t^B}{\lambda_{t'}^B}] \tag{4.1}$$

$$\lambda_t^{\gamma^B} = \lambda_{t'}^{\gamma^B}[(1-\gamma_{t'}^B)(1-\rho)(1-\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + (1-(1-\gamma_{t'}^B)(1-\rho)(1-\alpha))\frac{\lambda_t^B}{\lambda_{t'}^B}] \tag{4.2}$$

En résumé, le spread de la pool est $\Delta = r^B - r^S$ et deux facteurs peuvent augmenter le spread pair à pair de Morpho : le facteur de réserve et les deltas. La différence $r^B - r^\alpha$ est notée Δ^B dans le diagramme suivant et le facteur de réserve ρ peut être considéré comme une proportion de l'amélioration qui est prise sur l'emprunt et l'offre côté, sans tenir compte des deltas. De même, les deltas pair à pair peuvent être considérés comme la proportion du montant pair à pair qui se trouve réellement sur la pool. On voit que ces deux facteurs sont multiplicatifs : la proportion de l'amélioration qui est perdue sur le taux r^{γ^B} est donnée par $1-(1-\rho)(1-\gamma^B)$. On peut ainsi voir le facteur de réserve comme une coupe sur l'amélioration par rapport à la pool, en tenant compte des deltas.



Le théorème suivant nous donne une intuition sur la façon dont les deltas évoluent s'ils ne sont pas réduits par de nouvelles interactions avec le protocole.

Théorème 4.3.0.1. *Entre deux mises à jour sans autre interaction sur Morpho, la part du delta de prêt γ^S diminue, et la part du delta d'emprunt γ^B augmente.*

Démonstration. Nous utilisons les notations et formules précédentes 4.1 et 4.2. Puisque nous

examinons les parts delta entre deux mises à jour sans interaction avec Morpho entre les deux, les quantités δ^S, δ^B, S^{P2P} et B^{P2P} sont constants. Ainsi, nous n'avons qu'à montrer que :

$$\frac{\lambda_{t'}^S}{\lambda_{t'}^{\gamma^S}} \cdot \frac{\lambda_t^{\gamma^S}}{\lambda_t^S} \geq 1$$
$$\frac{\lambda_{t'}^B}{\lambda_{t'}^{\gamma^B}} \cdot \frac{\lambda_t^{\gamma^B}}{\lambda_t^B} \leq 1$$

Nous avons

$$\begin{aligned}\frac{\lambda_{t'}^S}{\lambda_{t'}^{\gamma^S}} \cdot \frac{\lambda_t^{\gamma^S}}{\lambda_t^S} &= \frac{\lambda_{t'}^S}{\lambda_t^S} \cdot [(1-\gamma_{t'}^B)(1-\rho)(1-\alpha)\frac{\lambda_t^S}{\lambda_{t'}^S} + (1-(1-\gamma_{t'}^B)(1-\rho)(1-\alpha))\frac{\lambda_t^B}{\lambda_{t'}^B}] \\ &= (1-\gamma_{t'}^B)(1-\rho)(1-\alpha) + (1-(1-\gamma_{t'}^B)(1-\rho)(1-\alpha)) \cdot \frac{\lambda_t^B}{\lambda_{t'}^B} \cdot \frac{\lambda_{t'}^S}{\lambda_t^S}\end{aligned}$$

Puisque l'indice d'emprunt croît plus rapidement sur le pool que l'indice d'offre, nous avons $\frac{\lambda_t^B}{\lambda_{t'}^B} \cdot \frac{\lambda_{t'}^S}{\lambda_t^S} \geq 1$. Cela implique que

$$\frac{\lambda_{t'}^S}{\lambda_{t'}^{\gamma^S}} \cdot \frac{\lambda_t^{\gamma^S}}{\lambda_t^S} \leq (1-\gamma_{t'}^B)(1-\rho)(1-\alpha) + (1-(1-\gamma_{t'}^B)(1-\rho)(1-\alpha)) = 1$$

□

4.3.1 Théorème de Non-liquidation

Avec l'ajout du mécanisme delta, les lemmes 3.1.2.1 et 3.0.4.2 ne tiennent plus. Nous les redéfinissons pour prouver à nouveau le théorème de non-liquidation.

Intégrité des positions sur la pool

Lemme 4.3.1.1. *(Intégrité des positions sur la pool) La position de Morpho sur la pool est toujours la somme des positions sur la pool de tous ses utilisateurs, plus les deltas.*

$$\forall \theta \in \Theta_{M_\psi}, s_\theta(\overline{M_\psi}) = \sum_u s_\theta^{Pool}(u) + \delta_\theta^S$$

$$\forall \theta \in \Theta_{M_\psi}, b_\theta(\overline{M_\psi}) = \sum_u b_\theta^{Pool}(u) + \delta_\theta^B$$

Démonstration. Une preuve similaire à celle du lemme 3.1.2.1 s'applique ici aussi. □

Innégalité pair à pair

Lemme 4.3.1.2. *(Innégalité pair à pair) La somme des balances pair à pair des prêteurs moins le delta de prêt pair à pair est* ***toujours*** *inférieure ou égale à la somme des balances pair à pair des emprunteurs moins le delta d'emprunt pair à pair.*

$$\forall \theta \in \Theta_{M_\psi}, \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_S} - \delta_\theta^S \cdot \lambda_\theta^S \leq \sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_B} - \delta_\theta^B \cdot \lambda_\theta^B$$

Démonstration. Nous allons faire le même type de preuve que pour 3.1.2.2, par récurrence.

Situation initiale L'inégalité est clairement vérifiée au déploiement.

Récurrence Soit $\sigma_i \in \Sigma$ un état où l'inégalité est vérifiée. Soit une transition qui transite vers l'état $\sigma_r \in \Sigma$.

Nous traitons le cas où la transition est associée à l'appel de fonction à `withdraw`. À l'heure actuelle, nous ne considérons que le marché θ, car les autres sont intacts, de sorte que thêta peut être omis en indice. Soit $(\theta, u, x) \in D_{\texttt{withdraw}}$ l'entrée de cet appel de fonction.

Nous considérons d'abord la mise à jour des index et essayons de prouver que l'égalité tient toujours à l'état $\sigma_{r'}$, un état intermédiaire juste après la mise à jour des index. Notez que les variables δ^S et S^{P2P} ne changent pas lors de la mise à jour de l'index. Cela signifie que nous avons $\sigma_{r'}(\delta^S) = \sigma_i(\delta^S)$ et $\sigma_{r'}(S^{P2P}) = \sigma_i(S^{P2P})$.

Nous notons $\sigma_i(\gamma^S)$ pour :

$$\frac{\sigma_i(\delta^S) \cdot \sigma_i(\lambda^S)}{\sigma_i(S^{P2P}) \cdot \sigma_i(\lambda^{\gamma^S})}$$

Considérons d'abord cette relation :

$$\begin{aligned}
L &= \sum_u \sigma_{r'}(s_\theta^{P2P}(u)) \cdot \sigma_{r'}(\lambda_\theta^{\gamma_S}) - \sigma_{r'}(\delta_\theta^S) \cdot \sigma_{r'}(\lambda_\theta^S) \\
&= \sum_u \sigma_i(s_\theta^{P2P}(u)) \cdot \sigma_i(\lambda_\theta^{\gamma_S}) \cdot \frac{\sigma_{r'}(\lambda_\theta^{\gamma_S})}{\sigma_i(\lambda_\theta^{\gamma_S})} - \sigma_i(\delta_\theta^S) \cdot \sigma_i(\lambda_\theta^S) \cdot \frac{\sigma_{r'}(\lambda_\theta^S)}{\sigma_i(\lambda_\theta^S)} \\
&= \sum_u \sigma_i(s_\theta^{P2P}(u)) \cdot \sigma_i(\lambda_\theta^{\gamma_S}) \cdot \left(\frac{\sigma_{r'}(\lambda_\theta^{\gamma_S})}{\sigma_i(\lambda_\theta^{\gamma_S})} - \sigma_i(\gamma_\theta^S) \cdot \frac{\sigma_{r'}(\lambda_\theta^S)}{\sigma_i(\lambda_\theta^S)} \right)
\end{aligned}$$

De plus, on a :

$$\frac{\sigma_{r'}(\lambda_\theta^{\rho^S})}{\sigma_i(\lambda_\theta^{\rho^S})} = (1-(1-\rho_\theta^S)\alpha)\cdot\frac{\sigma_{r'}(\lambda_\theta^S)}{\sigma_i(\lambda_\theta^S)} + (1-\rho_\theta^S)\alpha\cdot\frac{\sigma_{r'}(\lambda_\theta^B)}{\sigma_i(\lambda_\theta^B)}$$

Pareillement, pour 4.1, on a :

$$\frac{\sigma_{r'}(\lambda_\theta^{\gamma^S})}{\sigma_i(\lambda_\theta^{\gamma^S})} = (1-(1-\sigma_i(\gamma_\theta^S))(1-\rho_\theta^S)\alpha)\cdot\frac{\sigma_{r'}(\lambda_\theta^S)}{\sigma_i(\lambda_\theta^S)} + (1-\sigma_i(\gamma_\theta^S))(1-\rho_\theta^S)\alpha\cdot\frac{\sigma_{r'}(\lambda_\theta^B)}{\sigma_i(\lambda_\theta^B)}$$

donc :

$$\frac{\sigma_{r'}(\lambda_\theta^{\gamma_S})}{\sigma_i(\lambda_\theta^{\gamma_S})} = (1-\sigma_i(\gamma_\theta^S))\cdot\frac{\sigma_{r'}(\lambda_\theta^{\rho_S})}{\sigma_i(\lambda_\theta^{\rho_S})} + \sigma_i(\gamma_\theta^S)\cdot\frac{\sigma_{r'}(\lambda_\theta^S)}{\sigma_i(\lambda_\theta^S)}$$

En utilisant cette dernière égalité, on a :

$$\begin{aligned} L &= \sum_u \sigma_i(s_\theta^{P2P}(u))\cdot\sigma_i(\lambda_\theta^{\gamma_S})\cdot(1-\sigma_i(\gamma_\theta^S))\cdot\frac{\sigma_{r'}(\lambda_\theta^{\rho_S})}{\sigma_i(\lambda_\theta^{\rho_S})} \\ &= \frac{\sigma_{r'}(\lambda_\theta^{\rho_S})}{\sigma_i(\lambda_\theta^{\rho_S})}\left(\sum_u \sigma_i(s_\theta^{P2P}(u))\cdot\sigma_i(\lambda_\theta^{\gamma_S}) - \sigma_i(\delta_\theta^S)\cdot\sigma_i(\lambda_\theta^S)\right) \end{aligned}$$

Pareillement :

$$\begin{aligned} R &= \sum_u \sigma_{r'}(b_\theta^{P2P}(u))\cdot\sigma_{r'}(\lambda_\theta^{\gamma_B}) - \sigma_{r'}(\delta_\theta^B)\cdot\sigma_{r'}(\lambda_\theta^B) \\ &= \frac{\sigma_{r'}(\lambda_\theta^{\rho_B})}{\sigma_i(\lambda_\theta^{\rho_B})}\left(\sum_u \sigma_i(b_\theta^{P2P}(u))\cdot\sigma_i(\lambda_\theta^{\gamma_B}) - \sigma_i(\delta_\theta^B)\cdot\sigma_i(\lambda_\theta^B)\right) \end{aligned}$$

De plus, nous remarquons que la croissance de l'indice de prêt pair à pair sans tenir compte des deltas est inférieure à la croissance de l'indice d'emprunt pair à pair :

$$\frac{\sigma_{r'}(\lambda^{\rho_S})}{\sigma_i(\lambda^{\rho_S})} \leq \frac{\sigma_{r'}(\lambda^{\rho_B})}{\sigma_i(\lambda^{\rho_B})}$$

Du fait que l'innégalité soit vérifiée à l'état σ_i :

$$\sum_u \sigma_{r'}(s_\theta^{P2P}(u))\cdot\sigma_{r'}(\lambda_\theta^{\gamma_S}) - \sigma_{r'}(\delta_\theta^S)\cdot\sigma_{r'}(\lambda_\theta^S) \leq \sum_u \sigma_{r'}(b_\theta^{P2P}(u))\cdot\sigma_{r'}(\lambda_\theta^{\gamma_B}) - \sigma_{r'}(\delta_\theta^B)\cdot\sigma_{r'}(\lambda_\theta^B)$$

Nous avons prouvé que l'inégalité tient à $\sigma_{r'}$ si elle tient à σ_i. Maintenant, nous allons nous concentrer sur σ_r, l'état à la fin de la transition. A partir de maintenant, on écrira λ^S pour $\sigma_r(\lambda_\theta^S) = \sigma_{r'}(\lambda_\theta^S)$, et on fait de même pour $\sigma_r(\lambda_\theta^B)$, $\sigma_r(\lambda_\theta^{\gamma_S})$ et $\sigma_r(\lambda_\theta^{\gamma_B})$.

Nous ne passons pas par l'étape **retrait de la pool**, car cela ne modifie pas l'équilibre pair à pair d'un utilisateur ni d'un delta.

Considérons l'étape **réduire le delta de prêt pair à pair**.

Soit x^{δ_S} le montant qui est réduit du delta dans l'unité sous-jacente (le premier `matched` dans le pseudo-code).

Le delta de prêt pair à pair est réduit de x^{δ_S}/λ^S, et le solde de prêt pair à pair de l'utilisateur est réduit de $x^{\delta_S}/\lambda^{\gamma_S}$.

Considérez également l'étape **promouvoir les prêteurs**.

Soit x^{μ_S} le montant total de `matchSuppliers` (le second `matched` dans le pseudo-code).

Le solde de prêt pair à pair de l'utilisateur est réduit de $x^{\mu_S}/\lambda^{\gamma_S}$.

Et la somme des balances des autres fournisseurs est augmentée de $x^{\mu_S}/\lambda^{\gamma_S}$.

Enfin, considérez l'étape **rétrograder les emprunteurs**.

Soit x^{μ_B} le montant total sans pair à pair en natif par `unmatchBorrowers` (`unmatched` dans le pseudo-code).

Soit x^{δ_B} le montant ajouté au delta d'emprunt en natif (`montant - unmatched` dans le pseudo-code).

Le solde de prêt pair à pair de l'utilisateur est réduit de $(x^{\mu_B} + x^{\delta_B})/\lambda^{\gamma_S}$, et la somme du pair à pair les balances d'emprunt sont réduits de $x^{\mu_B}/\lambda^{\gamma_B}$.

Le delta d'emprunt pair à pair est augmenté de x^{δ_B}/λ^B.

Enfin :

$$\begin{aligned}
L &= \sum_u \sigma_r(s^{P2P}(u)) \cdot \lambda^{\gamma_S} - \sigma_r(\delta^S) \cdot \lambda^S \\
&= \left(\sum_u \sigma_{r'}(s^{P2P}(u)) - \frac{x^{\delta_S}}{\lambda^{\gamma_S}} - \frac{x^{\mu_S}}{\lambda^{\gamma_S}} + \frac{x^{\mu_S}}{\lambda^{\gamma_S}} - \frac{(x^{\mu_B} + x^{\delta_B})}{\lambda^{\gamma_S}} \right) \cdot \lambda^{\gamma_S} - \\
&\quad \left(\sigma_{r'}(\delta^S) - \frac{x^{\delta_S}}{\lambda^S} \right) \cdot \lambda^S \\
&= \sum_u \sigma_{r'}(s^{P2P}(u)) \cdot \lambda^{\gamma_S} - \sigma_{r'}(\delta^S) \cdot \lambda^S - x^{\mu_B} - x^{\delta_B} \\
R &= \sum_u \sigma_r(b^{P2P}(u)) \cdot \lambda^{\gamma_B} - \sigma_r(\delta^B) \cdot \lambda^B \\
&= \left(\sum_u \sigma_{r'}(b^{P2P}(u)) - \frac{x^{\mu_B}}{\lambda^{\gamma_B}} \right) \cdot \lambda^{\gamma_B} - \left(\sigma_{r'}(\delta^B) + \frac{x^{\delta_B}}{\lambda^B} \right) \cdot \lambda^B \\
&= \sum_u \sigma_{r'}(b^{P2P}(u)) \cdot \lambda^{\gamma_B} - \sigma_{r'}(\delta^B) \cdot \lambda^B - x^{\mu_B} - x^{\delta_B}
\end{aligned}$$

Donc :

$$\begin{aligned}
&\sum_u \sigma_r(s^{P2P}(u)) \cdot \lambda^{\gamma_S} - \sigma_r(\delta^S) \cdot \sigma_r(\lambda^S) \\
\leq &\sum_u \sigma_r(b^{P2P}(u)) \cdot \lambda^{\gamma_B} - \sigma_r(\delta^B) \cdot \sigma_r(\lambda^B)
\end{aligned}$$

On pourrait faire les mêmes preuves pour toutes les autres fonctions externes et pour Γ_{tick}. □

On revient au théorème de non-liquidation 3.1.2.1, avec une preuve mise à jour prenant en compte le mécanisme delta.

Théorème 4.3.1.1 (Non-liquidation avec deltas)**.** *Si Morpho est équipé d'un système de liquidation fonctionnel, alors la position de Morpho sur la pool n'est pas liquidable, même en présence de deltas. Ainsi l'inégalité suivante tient toujours :*

$$\sum_\theta p_\theta \cdot b_\theta(\overline{M_\psi}) \cdot \lambda_\theta^B \leq \sum_\theta p_\theta \cdot F_\theta \cdot s_\theta(\overline{M_\psi}) \cdot \lambda_\theta^S$$

Démonstration. La preuve est similaire à celle sans le facteur de réserve et le mécanisme delta. Ce qui change, ce sont surtout les hypothèses.

D'après le lemme d'inégalité pair à pair 4.3.1.2, nous savons que, pour chaque jeton θ, l'inégalité suivante est :

$$\sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_B} - \delta_\theta^B \cdot \lambda_\theta^B \geq \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_S} - \delta_\theta^S \cdot \lambda_\theta^S$$

Nous obtenons également les égalités suivantes à partir du lemme d'intégrité de la position de la pool 4.3.1.1 :

$$b_\theta(\overline{M_\psi}) = \delta_\theta^B + \sum_u b_\theta^{Pool}(u)$$

$$s_\theta(\overline{M_\psi}) = \delta_\theta^S + \sum_u s_\theta^{Pool}(u)$$

Nous avons toujours l'inégalité suivante à partir des hypothèses du théorème :

$$\sum_\theta p_\theta \cdot \sum_u \omega_\theta^B(u) \leq \sum_\theta p_\theta \cdot F_\theta \cdot \sum_u \omega_\theta^S(u)$$

Appelons L et R le membre gauche et le membre droit de l'inégalité précédente, et réduisons-les. Pour L :

$$\begin{aligned}
L =& \sum_\theta p_\theta \cdot \sum_u \omega_\theta^B(u) \\
=& \sum_\theta p_\theta \cdot \sum_u (b_\theta^{Pool}(u) \cdot \lambda_\theta^B + b_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma^B}) \\
=& \sum_\theta p_\theta \cdot ((b_\theta(\overline{M_\psi}) - \delta_\theta^B) \cdot \lambda_\theta^B + \sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_B}) \\
=& \sum_\theta p_\theta \cdot b_\theta(\overline{M_\psi}) \cdot \lambda_\theta^B + \sum_\theta p_\theta \cdot (\sum_u b_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_B} - \delta_\theta^B \cdot \lambda_\theta^B) \\
\geq& \sum_\theta p_\theta \cdot b_\theta(\overline{M_\psi}) \cdot \lambda_\theta^B + \sum_\theta p_\theta \cdot (\sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_S} - \delta_\theta^S \cdot \lambda_\theta^S)
\end{aligned}$$

Et pour R :

$$\begin{aligned}
R =& \sum_\theta p_\theta \cdot F_\theta \cdot \sum_u \omega_\theta^S(u) \\
=& \sum_\theta p_\theta \cdot F_\theta \cdot \sum_u (s_\theta^{Pool}(u) \cdot \lambda_\theta^S + s_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_S}) \\
=& \sum_\theta p_\theta \cdot F_\theta \cdot ((s_\theta(\overline{M_\psi}) - \delta_\theta^S) \cdot \lambda_\theta^S + \sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_S}) \\
=& \sum_\theta p_\theta \cdot F_\theta \cdot s_\theta(\overline{M_\psi}) \cdot \lambda_\theta^S + \sum_\theta p_\theta \cdot F_\theta \cdot (\sum_u s_\theta^{P2P}(u) \cdot \lambda_\theta^{\gamma_S} - \delta_\theta^S \cdot \lambda_\theta^S)
\end{aligned}$$

Donc, parce que $F_\theta \leq 1$, on peut toujours conclure que :

$$\sum_\theta p_\theta \cdot b_\theta(\overline{M_\psi}) \cdot \lambda_\theta^B \leq \sum_\theta p_\theta \cdot F_\theta \cdot s_\theta(\overline{M_\psi}) \cdot \lambda_\theta^S$$

□

4.4 Calcul du paramètre N

L'introduction d'un hyperparamètre N dans l'algorithme de Mariage de Morpho a un impact significatif sur l'efficacité du protocole. Une valeur trop faible de N peut restreindre le nombre d'utilisateurs pairés dans une transaction, alors qu'une valeur trop élevée pourrait augmenter les coûts d'utilisation et potentiellement dissuader l'adoption du protocole.

4.4.1 Introduction du paramètre de gas

Contrairement à une itération fixe, chaque cycle dans l'algorithme de mariage peut engendrer un coût variable, dépendant de plusieurs facteurs tels que la position de l'utilisateur dans la structure de données, le montant impliqué et l'état actuel de la structure. C'est pourquoi nous avons opté pour une limite de consommation de gas plutôt qu'un nombre d'itérations.

Le choix du plafond de gas est déterminé par une multitude de facteurs, incluant le choix de la structure de données et les objectifs stratégiques liés au coût maximal acceptable pour les utilisateurs de Morpho. Pour le déploiement initial de Morpho, le gas maximum alloué à l'algorithme de Mariage a été fixé à 100000 pour les quatre types de transactions.

4.5 Augmentation artificielle des deltas

Les deltas, par nature, sont indépendants des utilisateurs et tirent parti de la mutualisation pour assurer un taux d'intérêt égal ou supérieur à celui de la pool. L'augmentation artificielle d'un delta permettrait de basculer instantanément tous les utilisateurs pairés sur la pool sous-jacente. Voici le processus :

- Morpho emprunte sur la pool jusqu'à atteindre sa capacité d'emprunt maximale, garantie par le théorème de non-liquidation.
- De façon atomique, Morpho redépose le montant emprunté sur la pool.
- Les deltas sont ajustés pour inclure le montant emprunté et déposé sur la pool.

Si la capacité d'emprunt est restreinte par rapport au montant total pairé, cette opération peut être répétée jusqu'à ce que le pairage soit complètement résorbé.

Bien que cette manipulation du delta puisse sembler contre-intuitive en réduisant le taux en pair à pair, elle offre la flexibilité de redéfinir le comportement du modèle sans pair à pair si nécessaire.

Conclusion

Nous avons exploré en détail le mécanisme des deltas dans Morpho, en introduisant de nouvelles variables définissant les marchés et en ajoutant une logique complexe à l'algorithme de Mariage. Il a été rigoureusement démontré que ces mécanismes innovants ne compromettent en rien les propriétés essentielles et les théorèmes sur lesquels repose Morpho, tout en offrant une flexibilité opérationnelle accrue et une efficacité améliorée.

À ce stade, Morpho est parfaitement configuré pour fonctionner de manière optimale sur la blockchain Ethereum. Les transactions restent économiques grâce à la gestion prudente du paramètre N, qui limite les coûts de gas tout en maximisant l'efficacité du pairage. Le défi consistait à équilibrer les coûts d'utilisation avec l'efficacité du protocole, afin de garantir un taux de pairage élevé sans compromettre l'accessibilité et la viabilité économique.

Dans la partie suivante de ce mémoire, nous nous concentrerons sur l'analyse de l'éfficacité de Morpho avec le delta par rapport aux autres protocoles à l'état de l'art, notament Aave [4]. L'objectif est d'évaluer l'efficacité de Morpho par rapport à Aave en termes de coûts opérationnels et de taux d'intérêt offerts aux utilisateurs. Nous étudierons également l'influence des deltas sur l'engagement des utilisateurs avec Morpho et sur l'économie générale du protocole. L'efficacité de l'algorithme de Mariage sera analysée en profondeur pour déterminer son impact sur la performance globale du protocole et sa capacité à fournir une amélioration de Pareto par rapport aux systèmes de prêts et d'emprunts existants sur Ethereum.

Enfin, nous proposerons une synthèse de ces résultats, en mettant en lumière les avantages concurrentiels de Morpho et en offrant des perspectives sur l'avenir des protocoles de prêt et d'emprunt dans l'espace de la finance décentralisée. Les enseignements tirés de cette étude serviront de base pour de futures améliorations et pourraient inspirer le développement de nouvelles fonctionnalités, positionnant Morpho comme un acteur innovant et prééminent sur le marché de la DeFi.

CHAPITRE 5 ÉTUDE COMPARATIVE ENTRE MORPHO AAVE V2 ET AAVE V2

Cette dernière partie se concentre sur une comparaison entre Morpho AaveV2, une instance de Morpho fonctionnant au-dessus des pools d'AaveV2, et AaveV2, un acteur bien établi dans le domaine de la finance décentralisée. Pour mener à bien cette analyse, nous avons développé deux approches : une simulation basée sur un jeu de données extraites d'AaveV2, et une évaluation du protocole Morpho AaveV2 en production sur Ethereum. Cette section comprend également une analyse comparative des coûts en gas entre Morpho et Aave.

5.0.1 Simulation de Morpho avec un Jeu de Données d'AaveV2

Notre première démarche implique la simulation de Morpho sans le mécanisme Delta, utilisant les données réelles issues d'AaveV2. Cette simulation vise à évaluer la capacité de Morpho à maintenir son efficacité sans Delta et à identifier les cas où le Delta devient nécessaire pour son opérationnalité à grande échelle.

5.0.2 Analyse des Intérêts Générés par Morpho AaveV2 en Production

La seconde démarche analyse Morpho AaveV2 en production, en utilisant un subgraph pour suivre en temps réel les améliorations des intérêts générés par le pair à pair. Ceci fournit une comparaison directe entre Morpho et les protocoles conventionnels en termes d'intérêts redistribués aux utilisateurs.

5.1 Simulation de Morpho sans Delta

Nous démontrerons, grâce à un jeu de données extraites d'Aave, que Morpho nécessite le Delta pour fonctionner correctement. Cette simulation est effectuée sans le mécanisme Delta.

5.1.1 Description

Notre objectif est de recueillir des statistiques sur la nécessité du Delta. Si même une seule transaction requiert un Delta, cela confirme son importance pour maintenir Morpho en tant qu'amélioration de Pareto vis-à-vis d'Aave. Nous analyserons les transactions échouant à cause de l'absence de Delta, en mettant l'accent sur leur montant. Seules les transactions de remboursement et de retrait peuvent créer un Delta.

5.1.2 Analyse des résultats

La distribution des montants des transactions analysées est récapitulée dans le tableau 5.2. Le jeu de données inclut 3366 utilisateurs uniques et 14195 transactions, avec une valeur de gas pour le Delta définie à 100000.

Type de transaction	**Nombre de transactions**	**Dont delta**
Remboursement	2108	0
Retrait	2261	16
Emprunt	5782	-
Dépot	4044	-
Total	**14195**	-

Tableau 5.1 Récapitulatif des transactions avec et sans delta

Notre simulation, menée avec le jeu de données extrait d'Aave, révèle que sur 8738 transactions de remboursement et de retrait, 32 auraient nécessité un Delta pour passer sur Morpho. Bien que ce nombre puisse sembler faible, il est essentiel pour assurer le bon fonctionnement de Morpho. Sans Delta, ces transactions auraient échoué, compromettant la promesse de Morpho d'offrir une amélioration constante sur Aave.

Statistique	**Valeur**
Compte	14195
Moyenne	68427.82
Écart-type	266920
Minimum	0.001071
25%	2455.12
Médiane (50%)	10000
75%	45000
Maximum	10000000

Tableau 5.2 Répartition des montants

L'analyse des données simulées, comme illustré dans le tableau 5.2, met en évidence la diversité des montants traités dans les transactions. Malgré cette variabilité, seule une fraction des transactions nécessite l'intervention du Delta, soulignant ainsi son rôle critique dans le maintien de la performance de Morpho.

5.2 Mesure de l'éfficacité de Morpho

Nous évaluons maintenant l'efficacité de Morpho en nous appuyant sur la formule énoncée dans le Chapitre 2 (formule 3.9). Au cours de notre simulation, l'efficacité de l'algorithme de Mariage a été calculée en tenant compte du nombre maximal d'itérations autorisées pour chaque type de transaction, comme le montre la figure 5.2.

5.2.1 Analyse des résultats

Il est notable que l'interruption de l'algorithme de Mariage n'entraîne pas de baisse substantielle de l'efficacité de Morpho. L'efficacité reste remarquablement proche de 100%. Qui plus est, puisque les diminutions d'efficacité sont rapidement compensées par les transactions suivantes, comme mentionné dans les ajustements précédemment apportés à l'algorithme de Mariage, l'efficacité se rétablit promptement à son niveau optimal.

Une analyse plus détaillée de l'efficacité en fonction du type de transaction révèle que les emprunts sont les principaux responsables des variations d'efficacité. Ainsi, l'arrêt de l'algorithme suite au nombre d'itérations est la cause principale de la réduction de l'efficacité. Cela indique que le prêteur suivant peut être immédiatement apparié, sans même attendre dans la file d'attente. Par ailleurs, l'emprunteur peut opter pour un temps de traitement plus long lors de son emprunt pour maximiser son appariement et bénéficier de taux d'intérêt plus avantageux.

Cette capacité de réaction de Morpho permet d'absorber rapidement toute inefficacité temporaire et assure la résilience du protocole face aux fluctuations de l'activité.

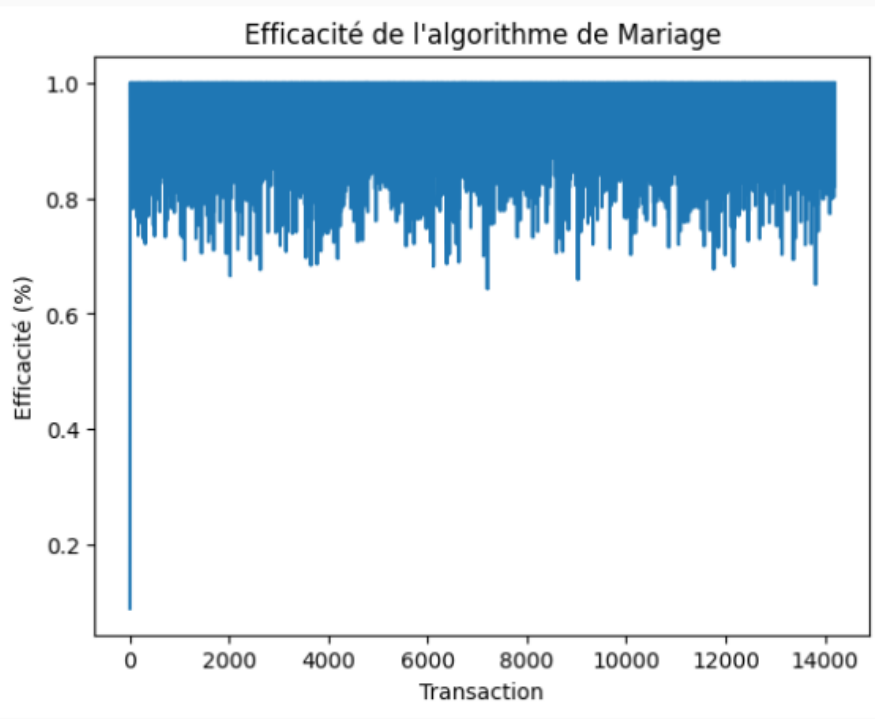


FIGURE 5.1 Éfficacité de l'Algorithme de Mariage, avec N=5 mariage par utilisateur max

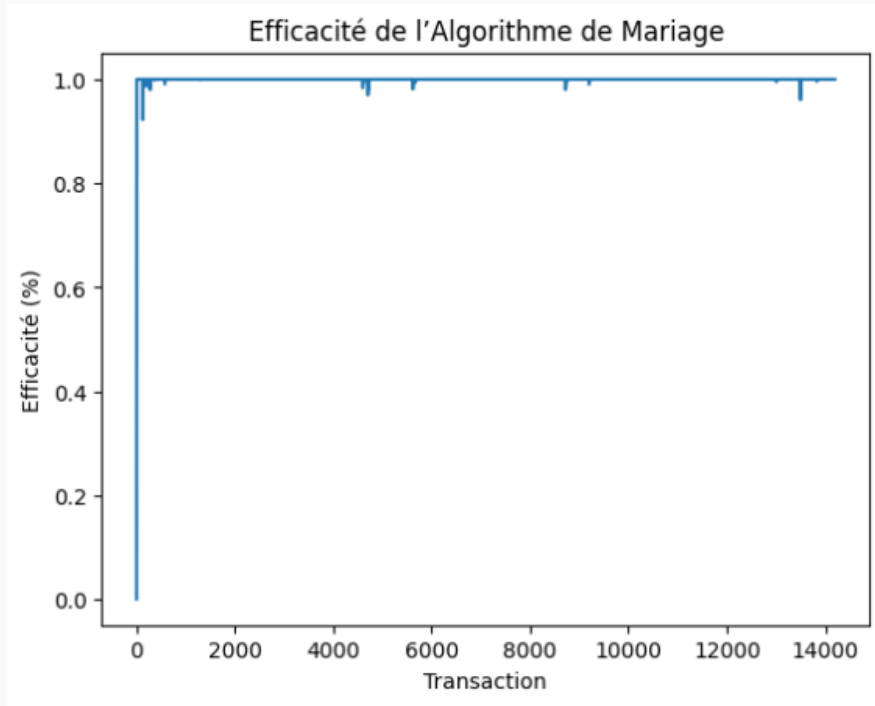


FIGURE 5.2 Éfficacité de l'Algorithme de Mariage

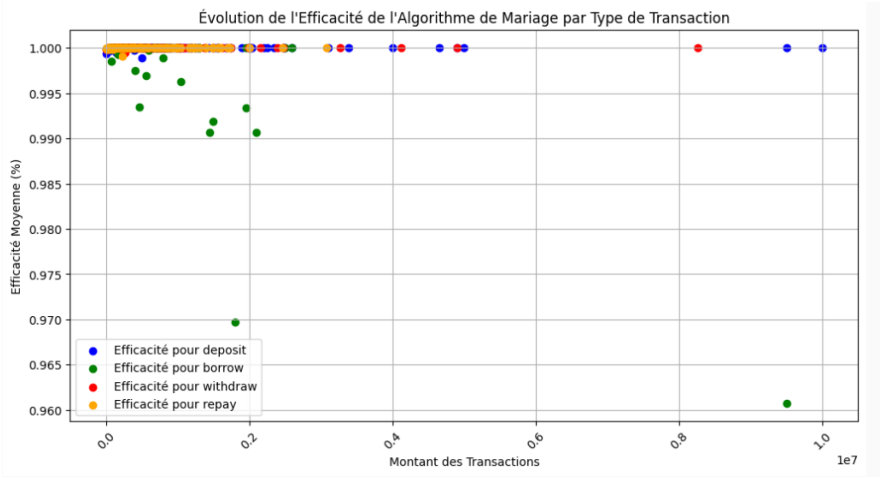


FIGURE 5.3 Éfficacité de l'Algorithme de Mariage par type de transaction

Ces résultats confirment la robustesse de Morpho face à une variété de scénarios transactionnels, renforçant son statut de protocole de prêt et d'emprunt hautement efficace et adaptable sur la blockchain Ethereum. On voit également qu'un modèle de mise en pair à pair avec une limite absolue du nombre de mise en pair à pair réduit notablement l'efficacité de l'algorithme. L'implémentation du delta permet donc d'améliorer l'efficacité de L'algorithme.

L'implémentation du mécanisme Delta se révèle être un complément essentiel au bon fonctionnement de Morpho, permettant au protocole de s'adapter aux exigences de la blockchain Ethereum. La capacité de Morpho à maintenir une efficacité proche de l'idéal, même avec un nombre d'itérations limité, souligne l'efficacité de son algorithme de Mariage et confirme son potentiel en tant que solution de prêt et d'emprunt de nouvelle génération dans l'écosystème de la finance décentralisée.

5.2.2 Transition vers la Comparaison entre Aave et Morpho

La section précédente a mis en lumière une efficacité presque optimale de Morpho grâce à son mécanisme de pair à pair. Toutefois, la question de l'impact réel de ce pair à pair sur les intérêts générés, comparativement à Aave, demeure. Pour répondre à cette interrogation, nous procéderons à la simulation des indices de pair à pair superposés à Aave, dans le but d'évaluer l'impact financier pour les utilisateurs optant pour Morpho. Il est crucial de noter que, dans la pratique, un tel modèle de superposition reste théorique car le taux en pair à pair et l'algorithme de matching influencent directement les taux d'intérêts. Ainsi, Morpho dépend fondamentalement de la pool sous-jacente, rendant son fonctionnement isolé impossible.

Cependant, chaque utilisateur trouve un intérêt individuel à migrer vers Morpho en raison

des avantages économiques personnels qu'il offre. C'est cette combinaison d'intérêt individuel et d'efficacité collective du pair à pair que nous allons analyser. Cette démarche mettra en exergue la valeur ajoutée de Morpho par rapport à Aave, tant en termes de coûts de transaction qu'en termes de bénéfices individuels pour les utilisateurs.

Nous aborderons maintenant une comparaison empirique entre Aave et Morpho, afin de démontrer, via des simulations basées sur des données réelles, comment l'adoption de Morpho se traduit par une amélioration tangible des intérêts perçus par les utilisateurs. Cette étude confirme l'intérêt de la proposition de valeur de Morpho et sa capacité à redéfinir les standards des protocoles de prêt et d'emprunt dans la finance décentralisée.

5.2.3 Analyse de l'Amélioration des Intérêts par le Pair à Pair sur Morpho AaveV2

Pour cette analyse, nous utilisons TheGraph [1], un outil d'analyse de données sur des transactions de blockchain publique, pour créer et déployer des subgraphs en AssemblyScript, permettant l'indexation et le stockage d'entités personnalisées lors de l'indexation des transactions.

Le subgraph de Morpho AaveV2 [2], déployé avec plus de 700 millions de dollars de dépôts, nous offre un aperçu des métriques suivantes :

- **_virtualScaledBorrow** : Balance des emprunts en pair à pair, indexée à la vitesse de la pool.
- **_virtualScaledSupply** : Balance des dépôts en pair à pair, indexée à la vitesse de la pool.
- **p2pSupplyInterestsImprovement** : Gain additionnel pour un prêteur quittant un marché pair à pair.
- **p2pBorrowInterestsImprovement** : Économie réalisée par un emprunteur lors du remboursement d'un emprunt en pair à pair.

Bien que centrée sur le marché USDC, cette analyse s'étend en temps réel à tous les marchés via le subgraph, dont le code source est accessible en open source sur Github [3].

La Figure 5.4 met en évidence la surperformance significative de Morpho en termes de rendement global. L'analyse, basée sur un protocole en production, assure la fiabilité des données utilisées.

1. https ://thegraph.com/
2. https ://etherscan.io/address/0x777777c9898d384f785ee44acfe945efdff5f3e0
3. https ://github.com/morpho-org/morpho-optimizers-subgraph

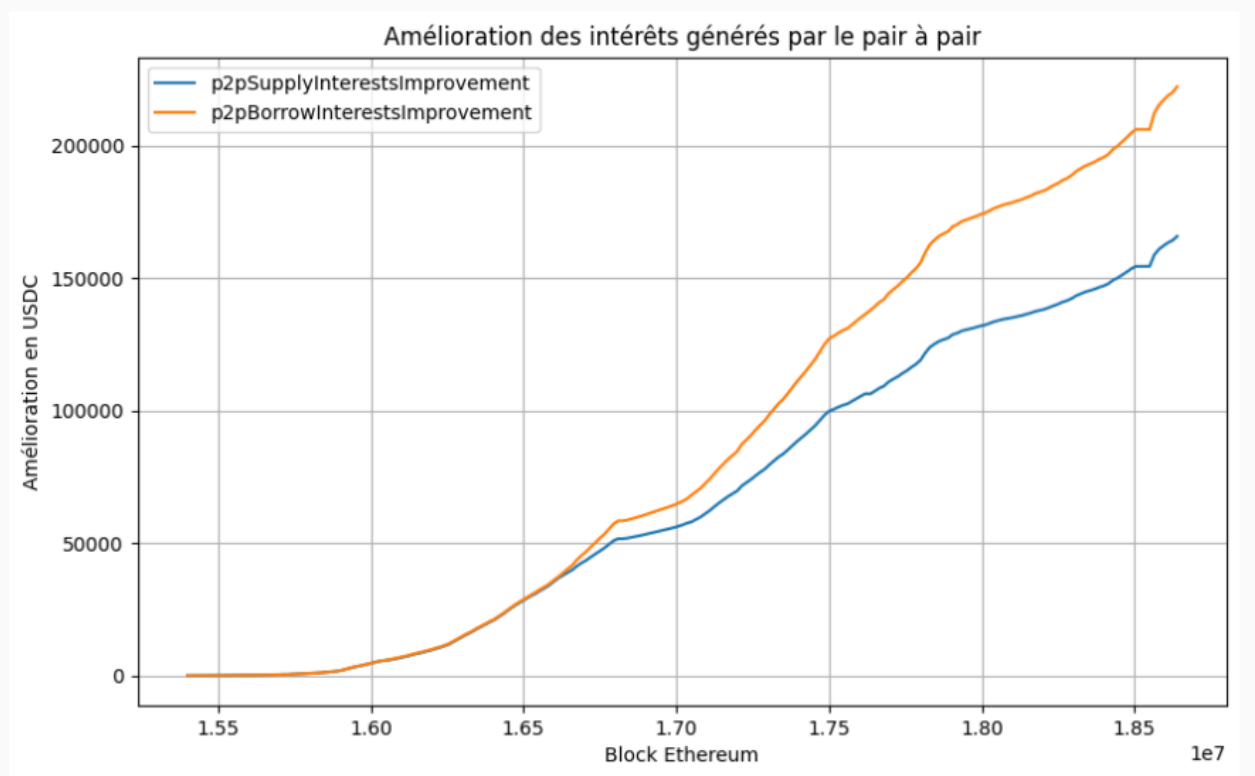


FIGURE 5.4 Amélioration des intérêts générés par le pair à pair sur Morpho AaveV2.

Il est essentiel de noter que l'instance de Morpho AaveV2 a été déployée avec le mécanisme Delta, témoignant d'une supériorité significative sur les protocoles de prêt et d'emprunt actuels en termes d'intérêts distribués. Cette efficacité accrue valide l'intégration du Delta au sein de Morpho, démontrant une nette surperformance par rapport à Aave.

5.2.4 Comparaison des taux d'intérêts

Les taux d'intérêt en pair à pair de Morpho sont étroitement liés aux taux de la pool. Ainsi, la performance de Morpho en termes de taux d'intérêt est fortement corrélée à celle de la pool sous-jacente, ici Aave. Comme démontré dans les chapitres précédents, Morpho offre des taux systématiquement plus avantageux que ceux de la pool sous-jacente, indépendamment du facteur de réserve, du curseur pair à pair et du delta. La Figure 5.5 illustre le taux en pair à pair de Morpho, sans frais et sans delta.

L'utilisation rapide du delta et sa résorption assurent une forte corrélation entre ce taux et le taux réel perçu par l'utilisateur en pair à pair.

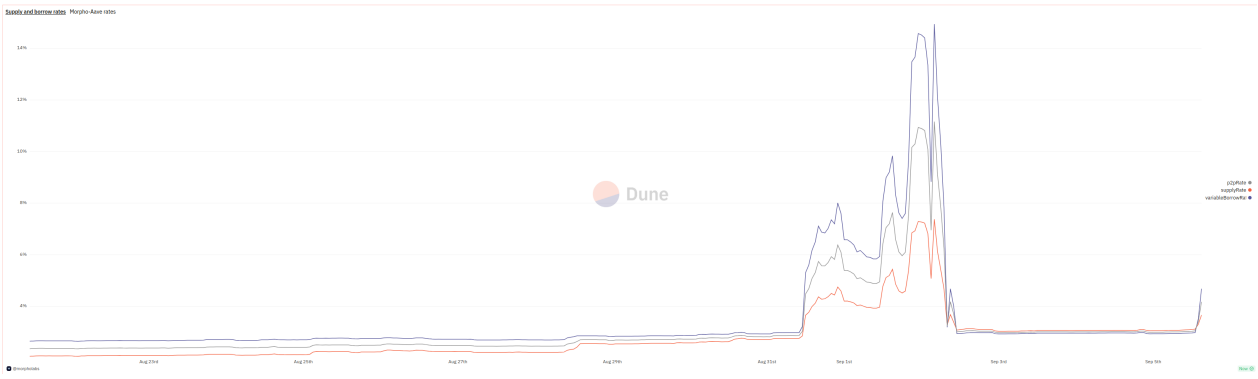


FIGURE 5.5 Taux de Aave et taux de Morpho

5.3 Comparaison du coût en gas

Étant une surcouche des protocoles de pool, Morpho présente un coût d'utilisation supérieur à celui des protocoles sous-jacents. Toutefois, ce surcoût en gas est principalement encouru lors de l'entrée ou de la sortie d'un marché, tandis que l'amélioration des taux est continue, motivant les utilisateurs à adopter Morpho. Néanmoins, il est crucial de maintenir ce coût dans des limites raisonnables pour encourager l'adoption du protocole. Nous allons examiner le coût moyen d'utilisation de Morpho et le comparer à celui de AaveV2.

L'analyse du jeu de données révèle un surcoût substantiel en gas pour Morpho, avec un coût triplé pour les dépôts, remboursements et retraits, comparativement à AaveV2. Ce surcoût s'explique par la complexité ajoutée par Morpho, incluant l'algorithme de Mariage et la double comptabilité au sein de Morpho et de la pool. Malgré ce surcoût, le coût d'entrée sur Morpho, d'environ 40 $, reste raisonnable pour les utilisateurs manipulant de gros montants, justifiant l'adoption de Morpho.

Type	**Gas sur Morpho**	**Gas sur Aave**	**Surcoût**
Emprunt	548298	367567	+149%
Dépot	353520	110253	+320%
Remboursement	358395	112876	+317%
Retrait	353884	108977	+324%

TABLEAU 5.3 coût en gas moyen sur Morpho

Conclusion

En conclusion, cette étude a permis de mettre en évidence l'importance du paramètre N dans le fonctionnement de l'algorithme de Mariage de Morpho et son impact direct sur la

performance et l'efficacité du protocole. La mise en œuvre de l'algorithme de Mariage avec une limite de gas maximale a montré qu'il est possible de concilier efficacité du pairage et maîtrise des coûts sur la blockchain Ethereum. L'analyse approfondie des simulations a révélé que Morpho peut offrir des taux d'intérêt compétitifs tout en garantissant des coûts de transaction raisonnables, élément crucial pour l'adoption et l'utilisation à long terme du protocole.

Les résultats obtenus ouvrent la voie à des recherches futures visant à affiner les mécanismes de pairage et à optimiser davantage l'algorithme de Mariage pour s'adapter à différentes conditions de marché. L'impact du Delta sur l'engagement des utilisateurs et sur l'ensemble de l'écosystème de la DeFi mérite une attention particulière pour assurer la pérennité et la croissance de Morpho dans le paysage des protocoles de prêt et d'emprunt.

En définitive, Morpho se positionne comme un protocole innovant capable de relever les défis techniques et économiques de la finance décentralisée sur Ethereum. Les enseignements tirés de cette analyse contribueront à la conception de systèmes de prêt et d'emprunt plus efficaces, transparents et accessibles, alignés sur les principes fondamentaux de la DeFi et sur les attentes des utilisateurs. La mise à l'échelle réussie de Morpho est un pas significatif vers la réalisation de ces objectifs et renforce la vision d'un avenir financier ouvert et démocratisé.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des Travaux

La présente étude a débuté par une revue de littérature sur les protocoles de prêt et d'emprunt dans l'écosystème de la finance décentralisée (DeFi), mettant en lumière les mécanismes fondamentaux régissant des plateformes telles qu'Aave et Compound. L'analyse a révélé l'importance d'une interaction efficace entre les utilisateurs et ces protocoles, ainsi que les défis associés à l'échelle et au coût des transactions sur Ethereum.

Dans un second temps, nous avons examiné Morpho sous toutes ses coutures, révélant un protocole innovant qui s'appuie sur un modèle de pool enrichi d'une couche de pair à pair. Cette combinaison vise à concilier les taux avantageux du pair à pair avec la liquidité et la robustesse des pools. L'intégration du Delta, en tant que mécanisme de rattrapage, a été mise en lumière, soulignant sa capacité à préserver l'intégrité des taux et la non-liquidabilité du protocole tout en garantissant sa mise à l'échelle.

La comparaison de Morpho avec Aave a permis de démontrer ses avantages en termes de mise à l'échelle du pair à pair et d'amélioration des taux d'intérêt pour les utilisateurs. La partie simulation a joué un rôle crucial en illustrant de manière empirique l'efficacité de Morpho par rapport au protocole AaveV2 sur le marché USDC, confirmant ainsi les hypothèses théoriques, et cela en toute situation.

6.2 Limitations de la Solution Proposée

Malgré les avancées significatives apportées par Morpho, certaines limitations doivent être soulignées. La dépendance accrue envers le modèle de pool sous-jacent limite la croissance de Morpho à celle de la pool, restreignant ainsi son potentiel d'optimisation indépendante. De surcroît, le coût d'une transaction sur Morpho sera inévitablement supérieur à celui d'une opération directe sur la pool, ce qui peut s'avérer un frein à l'adoption massive du protocole.

La rigidité du paramètre N dans l'algorithme de Mariage constitue également un enjeu. La détermination empirique de N, bien que fondée sur des données solides, pourrait bénéficier d'une approche plus dynamique et réactive aux conditions changeantes du marché.

Enfin, le mécanisme de Delta, bien qu'utile pour l'équilibrage des taux, requiert une attention particulière pour minimiser son impact sur les coûts et maximiser sa réactivité. L'exploration d'une délégation off-chain de l'algorithme de Mariage, avec une validation on-chain, pourrait

s'avérer une piste fructueuse pour contourner les contraintes de coût et d'échelle inhérentes à l'environnement d'Ethereum.

6.3 Impact de la Solution Proposée

Depuis juin 2022, trois instances de Morpho ont été déployées avec succès sur Ethereum : Morpho-CompoundV2 en mai 2022, Morpho-AaveV2 en août 2022, et Morpho-AaveV3 en mars 2023. Chacune de ces solutions a intégré le Delta et fonctionne sans interruption depuis leur lancement.

À ce jour, Morpho possède 1,2 milliards de dollars de dépôts [1], témoignant d'une croissance soutenue dans un contexte global marqué par des défis tant dans la finance traditionnelle que dans l'univers des cryptoactifs, fortement influencés par la situation géopolitique actuelle.

Les contrats intelligents de Morpho ont fait l'objet d'audits rigoureux par plus de dix entreprises spécialisées et reconnues dans l'audit de contrats intelligents. Plus de vingt audits de sécurité du code ont été menés, renforçant ainsi la confiance dans la robustesse de la plateforme. De plus, une prime de 555 555 dollars est offerte à quiconque découvrira une faille critique dans l'une des instances de Morpho. Cette initiative témoigne de l'engagement de Morpho en faveur de la sécurité et de la fiabilité.

Le Delta a démontré son efficacité de manière convaincante. Morpho, se positionnant désormais au troisième rang des protocoles de prêt et d'emprunt sur Ethereum, juste derrière Compound et Aave, illustre clairement que le modèle de prêt et d'emprunt en pair à pair a non seulement sa place, mais aussi un rôle crucial à jouer dans l'écosystème DeFi.

6.4 Améliorations futures et perspectives

L'avancée significative que représente Morpho dans l'écosystème de la finance décentralisée ouvre la voie à de nombreuses pistes d'amélioration et d'optimisation. La recherche continue sur la détermination de l'hyperparamètre N et sur la dynamisation de son ajustement en fonction des variations du marché et des comportements des utilisateurs est l'un des domaines prometteurs. L'objectif serait de développer une méthode permettant de calibrer N en temps réel, en prenant en compte des facteurs tels que la congestion du réseau, le coût du gas, et les tendances du marché, pour maximiser l'efficacité de l'algorithme de Mariage tout en minimisant les coûts pour les utilisateurs.

De même, une attention particulière pourrait être portée sur le curseur pair à pair, qui

1. Données en date de novembre 2023

détermine le taux d'intérêt dans les relations de prêt et d'emprunt pair à pair. Une approche plus sophistiquée pourrait consister à le moduler dynamiquement en fonction de l'offre et de la demande sur le marché, afin d'ajuster les taux d'intérêt de manière plus précise et réactive.

En ce qui concerne le Delta, des améliorations visant à réduire son coût sont envisageables. Cela pourrait passer par une optimisation des algorithmes on-chain ou une refonte de la logique de gestion des deltas pour réduire la charge computationnelle et le coût associé.

Une autre perspective intéressante serait de déléguer une partie des calculs de l'algorithme de Mariage off-chain. Cette externalisation permettrait de réduire les coûts de transaction sur la blockchain et de traiter des volumes de pairage plus importants. On-chain, un validateur du pairage s'assurerait de la véracité et de l'intégrité des matchings proposés, garantissant ainsi la sécurité et la confiance dans les opérations effectuées.

Ces améliorations futures visent à renforcer l'efficacité et l'attrait de Morpho en tant que protocole de prêt et d'emprunt sur Ethereum. L'objectif ultime est de créer un système plus agile, plus économique et mieux adapté aux besoins des utilisateurs, tout en conservant les avantages fondamentaux qui font l'intérêt de la finance décentralisée.

Conclusion générale

L'introduction de Morpho dans l'écosystème DeFi représente une étape importante vers l'amélioration de l'efficacité et de la mise à l'échelle des protocoles de prêt et d'emprunt sur Ethereum. Les avancées réalisées et les pistes d'amélioration identifiées sont cruciales pour le développement continu de la finance décentralisée. En dépit de certaines limitations inhérentes à la solution proposée, les perspectives d'optimisation et d'innovation offrent un horizon prometteur pour Morpho et pour l'ensemble de la DeFi. Les travaux futurs devront se concentrer sur l'amélioration des paramètres du protocole, la réduction des coûts et l'exploration de nouvelles méthodes pour une décentralisation accrue et une efficacité optimale comme évoquée dans le principe d'hyperstructures [24].

RÉFÉRENCES

[1] V. Buterin *et al.*, "Ethereum white paper," 2013.

[2] L. Gudgeon, S. Werner, D. Perez et W. J. Knottenbelt, "Defi protocols for loanable funds : Interest rates, liquidity and market efficiency," dans *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, ser. AFT '20. New York, NY, USA : Association for Computing Machinery, 2020, p. 92–112. [En ligne]. Disponible : https://doi.org/10.1145/3419614.3423254

[3] R. Leshner et G. Hayes, "Compound : The money market protocol (whitepaper)," Février 2019.

[4] A. Team, "Aave v1 protocol whitepaper," Janvier 2020.

[5] S. Kulechov. Ethlend whitepaper.

[6] V. D. Paul Frambot, Katia Babbar, "Morpho optimizers : Optimizing decentralized liquidity protocols (whitepaper)," Novembre 2021.

[7] M. Aquilina, J. Frost et A. Schrimpf, "Decentralised finance (defi) : A functional approach," *SSRN Transactions on Instrumentation and Measurement*, 2023.

[8] S. Nakamoto. (Juin 2023) Bitcoin : A peer-to-peer electronic cash system. [En ligne]. Disponible : https://bitcoin.org/bitcoin.pdf

[9] Bitcoin Community. Bitcoin improvements proposals. [En ligne]. Disponible : https://github.com/bitcoin/bips

[10] G. Wood, "Ethereum white paper," Juin 2023. [En ligne]. Disponible : https://ethereum.github.io/yellowpaper/paper.pdf

[11] V. Buterin, *Proof of Stake*. London : Seven Stories Press, 2022.

[12] Ethereum Community. Ethereum improvements proposals. [En ligne]. Disponible : https://github.com/ethereum/EIPs

[13] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu et G. Rosu, "Kevm : A complete formal semantics of the ethereum virtual machine," dans *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, p. 204–217.

[14] L. Gudgeon, S. Werner, D. Perez et W. J. Knottenbelt, "Decentralized finance : On blockchain- and smart contract-based financial markets," *arXiv preprint arXiv :2006.08825*, 2020.

[15] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz et W. J. Knottenbelt, "Sok : Decentralized finance (defi)," *arXiv preprint arXiv :2101.08778*, 2021.

[16] A. Klages-Mundt et D. Harz, “Insights into the issues and possible solutions of collateralized debt in decentralized finance,” dans *Workshop on Cryptocurrencies and Blockchains for Distributed Systems*. IEEE, 2020, p. 1–6.

[17] D. Perez, S. M. Werner, J. Xu et B. Livshits, “Liquidations : Defi on a knife-edge,” *arXiv preprint arXiv :2009.13235*, 2020.

[18] S. N. Cohen, M. Sabate-Vidales, L. Szpruch et M. Gontier Delaunay, “The paradox of adversarial liquidation in decentralised lending,” 2023.

[19] Euler Contributors, “Euler : an unstoppable liquidity protocol,” 2022.

[20] K. Qin, L. Zhou, P. Gamito, P. Jovanovic et A. Gervais, “An empirical study of defi liquidations : Incentives, risks, and instabilities,” dans *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC ’21. New York, NY, USA : Association for Computing Machinery, 2021, p. 336–350. [En ligne]. Disponible : https://doi.org/10.1145/3487552.3487811

[21] M. Bartoletti, B. Pes et S. Serusi, “Sok : Lending pools in decentralized finance,” *arXiv preprint arXiv :2006.13922*, 2020.

[22] D. L. Harz, “Security and efficiency of collateral in decentralized finance,” 2022. [En ligne]. Disponible : https://spiral.imperial.ac.uk/handle/10044/1/101394

[23] julian st, “Ethereum virtual machine (evm) : Documentation,” *https ://ethereum.org/en/developers/docs/evm/*.

[24] Jacob. Hyperstructures. [En ligne]. Disponible : https://jacob.energy/hyperstructures.html