Projet Algorithmes distribués avancer Article 2

Realiser par: EL HADDAD Roy
BEN SGHAIER Mohammed Ameur
DARDOR Rochdi
Presenter a : Madame PILARD Laurance
Madame COHEN Johanne
Monsieur SOHIER Devan

Table des matières

Introduction

Dans les systèmes distribués, la tolérance aux fautes et la capacité d'adaptation à un environnement dynamique constituent des exigences fondamentales pour la fiabilité et la robustesse. L'un des paradigmes clés pour répondre à ces contraintes est celui de l'auto-stabilisation.
L'article "A Self-stabilizing Weighted Matching Algorithm" de Fredrik Manne et Morten Mjelde, publié en 2007, s'inscrit dans cette thématique en proposant un algorithme auto-stabilisant pour le problème du couplage pondéré dans un graphe distribué. Ce problème consiste à sélectionner un sous-ensemble d'arêtes sans sommets communs (un matching), de manière à maximiser la somme des poids des arêtes choisies. Contrairement au cas non pondéré, la version pondérée présente des défis supplémentaires, notamment en ce qui concerne l'approximation et la stabilité du résultat.
Jusqu'à cette publication, les travaux existants en auto-stabilisation s'étaient concentrés sur le matching maximal non pondéré, pour lequel une solution simple greedy permettait déjà une approximation de facteur ½. Or, dans le cas pondéré, un matching maximal peut être très éloigné de l'optimal. Il était donc nécessaire de concevoir un nouvel algorithme spécifiquement adapté.
L'algorithme présenté dans l'article se distingue par sa simplicité, son efficacité locale, et sa convergence garantie sous plusieurs modèles d'exécution (daemon juste, adversarial, synchrone, etc.). Il offre une approximation constante de la solution optimale (facteur ½), avec une complexité mémoire réduite : seulement deux variables locales par nœud.
Dans ce rapport, nous allons analyser en profondeur cet algorithme, en mobilisant les notions abordées dans le cours "Algorithmique distribuée avancée". Nous commencerons par rappeler les concepts clés, avant de détailler l'algorithme, d'illustrer son fonctionnement par des exemples concrets, d'analyser ses garanties de convergence et d'efficacité, et enfin d'examiner ses apports théoriques et pratiques dans le champ des systèmes distribués.

Rappels de cours

1. Self-stabilisation (auto-stabilisation)

Un algorithme **auto-stabilisant** est capable de **converger vers une configuration correcte (état légitime)** depuis n'importe quelle configuration initiale, appelée **état non-légitime**.

- Cette propriété le rend **résilient aux fautes transitoires** (pannes, redémarrages, modifications topologiques).
- Le système **s'auto-répare** sans intervention humaine ou réinitialisation.

Dans le contexte distribué, cela signifie que chaque **nœud agit localement**, et que la stabilisation se fait de manière **décentralisée**, au fil des activations successives des règles locales.

2. États légitimes et non-légitimes

Un **état légitime** est une configuration globale du système dans laquelle :

- Aucun nœud n'est **privilégié** (c'est-à-dire, aucun ne peut appliquer de règle locale).
- Le résultat collectif satisfait la **spécification globale** du problème (ici, un couplage pondéré cohérent).

Un **état non-légitime** est toute autre configuration, y compris :

- Des couplages partiels non mutuels (exemple : $m_v = u$ mais $m_u \neq v$).
- Des valeurs de poids erronées (exemple : $h_v \neq w(v, m_v)$).
- Des conflits (plusieurs arêtes incidentes sélectionnées).

L'algorithme étudié garantit que **depuis n'importe quel état non-légitime**, une suite finie d'activations permet d'atteindre un état légitime.

3. Couplage maximal et couplage maximum

Dans un graphe non orienté un couplage est :

- Un ensemble d'arêtes sans sommets communs.
- Un **couplage maximal** est un couplage auquel **aucune autre arête** ne peut être ajoutée sans violer cette propriété.
- Un **couplage maximum** est un couplage de **taille ou poids maximal** parmi tous les couplages possibles.

Tous les couplages maximums sont maximaux, mais l'inverse n'est pas vrai.

Dans le **cas pondéré**, un couplage maximal peut être très **sous-optimal** en termes de poids total. C'est pourquoi l'algorithme de Manne & Mjelde vise non pas la maximalité mais une **approche approximative du couplage maximum pondéré**.

4. Matching pondéré et matching non pondéré

- Dans le cas **non pondéré**, tout matching maximal constitue automatiquement une ½-approximation du matching maximum.
- Dans le cas **pondéré**, cette propriété ne tient plus : un matching maximal peut être arbitrairement mauvais en termes de poids total.

Cela rend le problème du matching pondéré auto-stabilisant beaucoup plus difficile et justifie la recherche d'approches dédiées.

Objectifs et apports de l'article

1. Problématique abordée

L'article de Fredrik Manne et Morten Mjelde s'attaque à un **problème fondamental** en algorithmique distribuée : comment construire de manière **auto-stabilisée** un **matching pondéré** dans un graphe non orienté et réparti, tout en maximisant le poids total des arêtes sélectionnées ?

Plus précisément, il s'agit de trouver un **algorithme distribué** dans lequel chaque nœud, ne connaissant que ses voisins directs, puisse participer à la construction collective d'un **couplage pondéré**, en convergeant vers une **configuration légitime**, c'est-à-dire un état stable où les appariements sont cohérents, sans conflit, et de bonne qualité.

2. Constat : un manque dans l'état de l'art

Avant cette publication, plusieurs travaux proposaient des **algorithmes auto-stabilisants pour le matching non pondéré** :

- Hsu et Huang (1992) : premier algorithme avec borne O(n^3)
- Tel (1994) et Hedetniemi et al. (2001) : améliorations à O(n^3)puis O(m)
- Goddard, Gradinariu, Chattopadhyay : versions synchrones, aléatoires ou spécialisées

Cependant, **aucune solution n'existait pour le cas pondéré**, alors que celui-ci est **plus difficile** :

- Un **couplage maximal pondéré** peut être **très éloigné** du couplage **maximum** en termes de poids total.
- Il n'existe pas de règle greedy triviale garantissant une bonne approximation pondérée dans un contexte distribué.
- La gestion des états non-légitimes est plus complexe, car il faut corriger à la fois les paires incohérentes et les erreurs de poids.

3. Ce que propose l'article

L'article présente **le premier algorithme auto-stabilisant** capable de produire une **½-approximation du matching pondéré maximal**, avec les caractéristiques suivantes :

- **Modèle distribué** (chaque nœud connaît uniquement ses voisins et les poids des arêtes incidentes).

- **Aucune initialisation requise** : l'algorithme peut partir d'un **état non-légitime arbitraire**.
- Il converge toujours vers un **état légitime**, défini par :
  - l'absence de nœuds privilégiés ;
  - un ensemble d'arêtes mutuellement validées ;
  - la cohérence des poids enregistrés localement.

- **Deux variables locales par nœud seulement** :
  - m_v : voisin proposé pour le matching
  - h_v : poids du matching proposé

- **Simplicité extrême** :
  - Une seule fonction : $BestMatch()$
  - Une seule règle : $SetMatch$

- **Convergence prouvée** sous différents daemons :
  - Adversarial : $O(3^n)$ étapes dans le pire cas
  - Fair : $2 \cdot |M| + 1$ rounds (linéaire en nombre de nœuds)

4. Apport principal : combinaison de simplicité, résilience et garantie

L'innovation de l'article repose sur une **conjonction rare** dans les algorithmes distribués :

- **Garantie d'approximation** (½ du poids optimal)
- **Preuves de correction et de convergence**
- **Gestion explicite des états non-légitimes**, assurant leur correction automatique
- **Stabilisation vers un état légitime** dans un environnement asynchrone
- **Robustesse intrinsèque aux fautes**, grâce au paradigme auto-stabilisant

Présentation détaillée de l'algorithme

1. Modèle de graphe

L'algorithme s'exécute sur un **graphe non orienté pondéré** :

- $G = (V, E)$ avec :
  - $|V| = n$ sommets.
  - $|E| = m$ arêtes.
  - un **poids strictement positif** $w(u, v) > 0$ pour chaque arête $(u, v) \in E$

Chaque nœud $v \in V$:

- connaît ses voisins $N(v)$ et les poids associés aux arêtes incidentes.
- possède un **identifiant unique** ID_v (localement comparable).
- agit indépendamment, sans synchronisation globale.

2. Ordre total sur les arêtes

Pour éliminer toute ambiguïté dans les choix, les auteurs définissent une notion de **poids effectif** :

$$w_{eff}(v,u) = (w(v,u), max(ID_v, IDu), min(ID_v, ID_u))$$

Cette représentation lexicographique impose un **ordre total** entre toutes les arêtes incidentes à un nœud :

- Cela permet à chaque nœud de déterminer de façon **déterministe** quelle est la meilleure arête disponible.
- Elle est essentielle à la **définition d'un état légitime unique**.

3. Variables locales et définitions d'état

Chaque nœud v maintient deux variables :

- m_v : identifiant du voisin proposé pour le matching (ou null).
- h_v : poids enregistré pour cette proposition (ou 0 si y'en a pas eu de match).

À partir de ces variables, on peut définir :

- Un **état légitime** localement : si $m_v = u \; et \; m_v = v$, et que $h_v = w(v,u)$ et $h_u = w(u,v)$.

- Un **état non-légitime** : toute configuration où ces conditions ne sont pas remplies :

  - couplage unilatéral (exemple : $m_v = u$ mais $m_v \neq v$).
  - poids incorrect.
  - instabilité de la sélection.

Ces états non-légitimes seront corrigés par l'application de règles locales.

4. Calcul du voisinage admissible :

Chaque nœud peut à tout moment calculer :

$$N'(v) = \{u \in N(v) \mid w(v,u) \geq h_u\}$$

Ce sont les voisins pour lesquels **un appariement avec** v pourrait être **au moins aussi avantageux** que leur appariement actuel. Cela permet à v d'**évaluer ses opportunités** de manière locale.

5. Fonction de sélection :

$BestMatch(v)$:

$$return\ u:\ max_{u \in N'(v) \cup \{null\}} w(v,u)$$

Retourne le **meilleur candidat** dans $N'(v)$, c'est-à-dire la meilleure arête disponible pour v.

Si aucun nœud n'est admissible, retourne null.

6. Règle unique :

$SetMatch$:

$if\ m\_v\ \neq\ BestMatch(v)$ or $h\_v\ \neq\ w(v,\ m\_v)$:

$m\_v := BestMatch(v)$

$h\_v :=\ w(v,\ m\_v)$

Cette règle **corrige localement** les états non-légitimes en mettant à jour le voisin proposé si nécessaire et en corrigeant le poids associé.
C'est l'unique règle utilisée dans l'algorithme. Sa répétition dans le temps permet la **correction progressive des erreurs locales** et la **convergence vers une configuration globale légitime**.

7. Mécanisme global de stabilisation

Le système évolue par l'activation successive de la règle $SetMatch$ sur les **nœuds privilégiés**, c'est-à-dire ceux pour lesquels la condition est vraie.

Au fil des activations :

- Les propositions unilatérales sont éliminées.
- Les poids erronés sont corrigés.
- Les appariements s'organisent selon les meilleures arêtes disponibles.

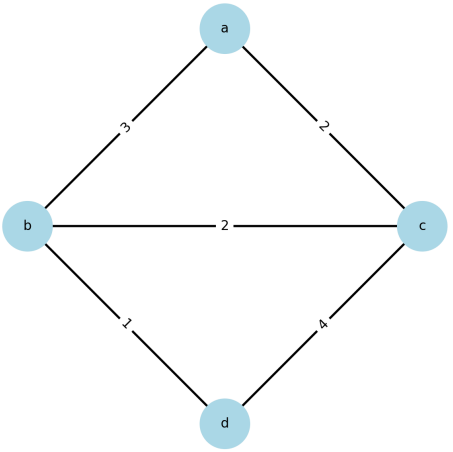
Quand **plus aucun nœud n'est privilégié**, on atteint un **état légitime global** :

- Chaque arête du matching est **mutuelle**.
- Aucun nœud ne peut améliorer son appariement.
- Les poids locaux sont cohérents.

Illustration complète avec exemple

Dans cette section, nous allons illustrer le fonctionnement de l'algorithme à l'aide d'un petit graphe de quatre nœuds, afin de suivre concrètement comment le système évolue depuis un **état non-légitime** vers un **état légitime stable**, en passant par des configurations intermédiaires.

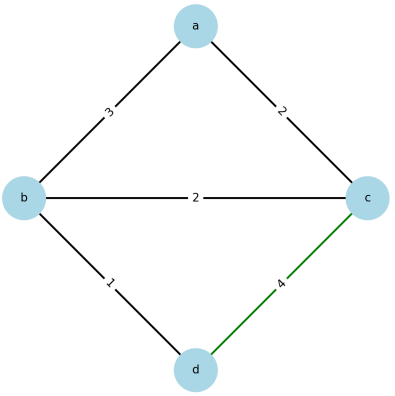
Étape 1 : Configuration initiale aléatoire :



Dans cet état non-légitime :

- Les variables locales des nœuds sont incohérentes.
- Aucun appariement mutuel n'a encore été formé.
- Chaque nœud peut être privilégié, et appliquer la règle $SetMatch$.

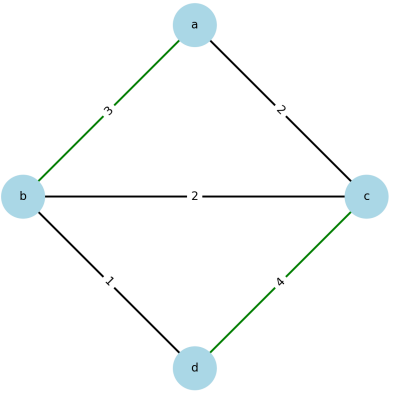
Étape 2 : Premier matching partiel :



Ici, c et d se reconnaissent mutuellement comme **meilleurs partenaires** selon $BestMatch$:

- Ils s'apparient sur l'arête de poids 4, la plus favorable.
- Leur état devient légitime entre eux.
- Les autres nœuds (a, b) restent instables ou célibataires.

Étape 3 : Matching final stable :



Enfin, a et b se matchent à leur tour :

- a détecte que b est admissible selon son $N'(a)$et vice versa.
- Les deux s'apparient sur l'arête de poids 3.
- Le système atteint une **configuration globale légitime** :
  - Plus aucun nœud n'est privilégié.
  - Tous les appariements sont mutuels et stables.
  - Le poids total du matching est de **7**, ce qui est proche de l'optimum.

Analyse de correction et de stabilisation

1. Mécanisme de stabilisation

L'algorithme repose sur une unique règle locale ($SetMatch$) qui permet, à chaque activation d'un nœud privilégié, de corriger son état. Cette correction est **strictement locale** (se base uniquement sur le voisinage direct), réduit progressivement le nombre de conflits et d'incohérences, converge toujours vers un état stable.

Chaque activation réduit un peu plus l'**instabilité locale**, jusqu'à atteindre une **configuration globalement légitime**.

2. Validité du matching

À l'état stable, pour toute arête $(v, u) \in M$:

- $m_v = u,\ m_u = v$
- $h_v = w(v, u),\ h_u = w(u, v)$

Ainsi, le matching final est constitué uniquement **d'arêtes mutuelles valides**.

3. Correction

Si $m_v = u$ mais $m_u \neq v$, alors soit v soit u est privilégié, donc va corriger son état.

Cela garantit que les propositions unilatérales **ne peuvent pas persister** dans une configuration stable et que le système évolue forcément vers un **matching mutuel**.

4. Optimalité

Théorème :

Le matching produit par l'algorithme est toujours une **½-approximation** du matching pondéré optimal M^* :

$$w(M) \geq \frac{1}{2} w(M^*)$$

Cela signifie que le poids total du matching final est **au moins la moitié** de celui du meilleur matching possible.

La preuve :

- Chaque arête de M^* est "couverte" par une arête dans M de poids équivalent ou supérieur,
- Chaque arête de M ne couvre au plus que 2 arêtes de M^*,

Donc, en sommant, on obtient la borne : $w(M^*) \leq 2 \cdot w(M) \Rightarrow w(M) \geq \frac{1}{2} w(M^*)$

5. Convergence

Modèle adversarial :

- Le système peut stabiliser **en** $O(3^n)$**étapes** dans le pire cas.
- Ce cas extrême correspond à un daemon très défavorable.

Modèle juste (fair daemon) :

- Le système stabilise **en au plus** $2 \cdot |M| + 1$ **rounds**,
- Puisque chaque appariement mutuel se stabilise en ≤ 2 tours.

Complexité et perspectives

L'algorithme de Manne & Mjelde a été conçu pour fonctionner dans un environnement **asynchrone, distribué et auto-stabilisant**. Ses performances dépendent du **modèle d'ordonnancement** (daemon) utilisé. Les auteurs analysent deux scénarios classiques : le daemon **adversarial** (pire cas) et le daemon **juste** (équitable).

1. Complexité dans le modèle adversarial

Dans ce modèle, le daemon peut retarder certains nœuds et orchestrer les activations de manière à ralentir la stabilisation. **Résultat** : Le système converge en $O(3^n)$ étapes dans le pire des cas.

Ce cas représente un environnement extrême, voire théorique et il démontre la **résilience** de l'algorithme : la stabilisation est **garantie**, même si très lente.

2. Complexité dans le modèle juste (fair daemon)

Dans un système équitable, chaque nœud privilégié finit par être activé. **Résultat** : Le système converge en au plus $2 \cdot |M| + 1$ **et** $|M|$ est la taille du matching finale (au plus $\frac{n}{2}$). Donc la complexité est **linéaire en nombre de nœuds** et le système converge rapidement dans des environnements réels ou modérément synchrones.

3. Couplage pondéré vs couplage maximal

L'algorithme ne cherche pas un **matching maximal** (où aucune arête ne peut être ajoutée), mais un **matching pondéré de qualité**, c'est-à-dire **proche du maximum de poids**.

- Cela évite de stabiliser sur des matchings lourds en taille mais **faibles en poids**.
- L'approximation ½ garantit une **qualité globale** en termes de gain pondéré, ce qui est souvent plus important dans les applications pratiques (qualité de lien, bande passante, etc.).

4. Perspectives et pistes de généralisation

Matching au-delà de ½ :

Les auteurs évoquent l'idée d'améliorer l'approximation en **détectant localement des chemins augmentants** :

- Un matching sans chemin augmentant de longueur ≤ 3 peut atteindre une **approximation de** $\frac{2}{3}$.
- Le défi est de concevoir un algorithme local, auto-stabilisant, capable d'intégrer ce raisonnement.

Généralisation à d'autres problèmes

La structure de l'algorithme pourrait inspirer des solutions pour :

- la **couverture minimale pondérée**.
- le **maximum independent set pondéré**.
- la **coloration pondérée auto-stabilisante**.

Modèles dynamiques

Étendre l'algorithme à des **graphes évolutifs** (où des arêtes apparaissent/disparaissent) permettrait de traiter les **réseaux mobiles**, les **systèmes intermittents** et les **environnements tolérants aux pannes** en temps réel.

Conclusion

L'article *"A Self-stabilizing Weighted Matching Algorithm"* de Fredrik Manne et Morten Mjelde constitue une avancée majeure en algorithmique distribuée, en proposant une solution simple mais puissante au problème du couplage pondéré dans un environnement asynchrone. À travers une approche auto-stabilisante, l'algorithme garantit la convergence vers un état légitime, sans dépendre d'une configuration initiale particulière ni d'un ordonnancement favorable.

L'intérêt de cette solution réside autant dans sa rigueur théorique que dans son applicabilité. Le système converge toujours, même depuis un état non-légitime, vers une configuration stable où les appariements sont mutuellement validés et localement optimaux. L'approximation de facteur ½ du matching optimal démontre qu'il est possible d'obtenir une solution de qualité tout en respectant les contraintes strictes de décentralisation.

Sur le plan pédagogique, cet article illustre parfaitement les notions abordées en cours, notamment la self-stabilisation, les états légitimes et non-légitimes, ainsi que la distinction entre couplage maximal et maximum. Il offre un cadre clair pour comprendre comment des comportements locaux simples peuvent produire un effet global correct, même dans un système potentiellement chaotique.

En somme, ce travail allie élégance, efficacité et robustesse, en faisant un modèle exemplaire d'algorithme distribué tolérant aux fautes, et un excellent support d'apprentissage.

Références

Manne, F., & Mjelde, M. (2007). *A self-stabilizing weighted matching algorithm*. In SSS 2007: Stabilization, Safety, and Security of Distributed Systems (pp. 282–295). Springer. https://doi.org/10.1007/978-3-540-77008-6_21

Hsu, C.-H., & Huang, S.-T. (1992). *A self-stabilizing algorithm for maximal matching*. Information Processing Letters, 43(2), 77–81. https://doi.org/10.1016/0020-0190(92)90070-7

Tel, G. (1994). *An efficient self-stabilizing algorithm for maximal matching*. In Proceedings of the 8th International Workshop on Distributed Algorithms (pp. 203–215). Springer. https://doi.org/10.1007/BFb0017231

Hedetniemi, S. M., Hedetniemi, S. T., Jacobs, D. P., & Harris, F. C. (2001). *Self-stabilizing algorithms for maximal matching and maximal independent sets*. Computers & Mathematics with Applications, 42(8–9), 1251–1261. https://doi.org/10.1016/S0898-1221(01)00239-0