Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Stratégies d'optimisation de la mémoire pour la calcul d'applications linéaires et l'indexation de document partagés

(***Résumé étendu en français***)

THÈSE

présentée et soutenue publiquement le

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

M. Mumtaz AHMAD

Composition du jury

Rapporteurs :	Denis LUGIEZ	Professeur, Université de Provence, Marseille, FR
	Miki (Nicolas) HERMANN	CR (CNRS), École Polytechnique, Paris, FR
Examinateurs :	Eugeniusz Adam CICHON	Professeur, Université Henri Poincaré, Nancy, FR
	Siva ANANTHARAMAN	Professeur, Université d'Orléans, Orleans, FR
	Abdessamad IMINE	Maître de conférences, Université Nancy 2, FR
	Lhouari NOURINE	Professeur, Université Blaise Pascal, Aubière, FR
	Nacer BOUDJLIDA	Professeur, Université Henri Poincaré, Nancy, FR
	Serge BURCKEL	Maître de conférences, Université de la Réunion, FR

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Acknowledgments

The memories of my Ph.D will be held in the highest regards and the support of my benefactors in achieving this goal will be cherished. I am really thankful to Michaël Rusinowitch, Research Director, CASSIS Project, INRIA Research center, Nancy Grand Est, for his nice cooperation. His encouraging statements in pessimistic situations, and moral support helped me to look forward and face the challenges with confidence. His patronage remained as mixed qualities of love and hardworking. I am thankful to Eugeniusz Adam Cichon (Professor, University Henri Poincare, Nancy, France) to be the director of my thesis and for his nice cooperation. His encouraging statements with spreading smiles increased my confidence. The INRIA CASSIS Group has been an excellent source of feedback and I am grateful to all members of the Group.

The efforts of my collaborators have developed my research skills, for which I am truly grateful, particularly to Abdessamad Imine and Serge Burckel. I have also benefited from the wisdom of Mathieu Turuani, Veronique Cortier, Laurent Vigneron and Christophe Ringeissen who have provided me academic support with nice suggestions and advices as members of my research group. Working in CASSIS Group, I enjoyed friendly environment. Academic life at INRIA/LORIA Research center, Nancy, has been wonderful and many factors have influenced my stay.

Best good luck wishes and prayers of dkq, mum and family provided me strong support in increased heartbeat situation of finalizing the presentation and thesis. Enthusiastic presence of new colleagues on the day of defense, kept me in smile and I benefited by good wishes of friends and teachers. I would like to be thankful to all members of the jury that they accepted our request. Moreover, I am thankful to professor Jacques Mazoyer (University Claude Bernard Lyon 1, France) and Priyadarsi Nanda, (University of Technology, Sydney, Australia) for accepting to be jury members. Finally, I would like to thank to everyone who helped me with this thesis, to ones who know they did and to ones who do not.

Je consacre cette thèse
à mes
bienfaiteurs

Table des matières

Table des figures

Liste des Algorithmes

Introduction Générale

« Mathematical discoveries, small or great are never born of spontaneous generation. They always presuppose a soil seeded with preliminary knowledge and well prepared by labour, both conscious and subconscious ».

Henri Poincaré (1854 - 1912)

Chapitre 1

Introduction

Contents

1.1 Décomposition séquentielle des opérations

La croissance rapide de l'utilisation de la technologie informatique moderne augmente la demande de meilleures performances dans tous les domaines de l'informatique. Cette demande pour des performances toujours plus grandes a conduit à la croissance des performances matérielles et à l'évolution de l'architecture, ce qui entraîne des contraintes sur la technologie de compilateur et pour la communauté de recherche. Étant donné que les microprocesseurs haute performance sont au cœur de chaque ordinateur à usage général, depuis les serveurs, les PC de bureau et portables, jusqu'aux téléphones cellulaires comme l'i Phone, l'augmentation du rendement est considéré comme un défi permanent dans l'informatique. En informatique, pour le calcul, les opérations élémentaires sont effectuées sur des objets élémentaires, *e.g.*, un processeur 32 bits ne peut effectuer des opérations que sur 32 bits. Ainsi, toute transformation de la structure de données doit être décomposée en opérations successives sur 32 bits. Pour échanger le contenu des registres, la solution habituelle pour assurer la complétude du calcul est de faire des copies à partir des données initiales. Mais cette solution peut générer des erreurs de mémoire lorsque les structures sont trop grandes, ou du moins diminue les performances de calculs. En effet, ces opérations impliquant plusieurs registres dans un micro-processeur, comme un compilateur ou un circuit électronique, nécessitent de copier certains registres dans la mémoire cache ou en mémoire vive, avec une perte de vitesse, ou de dupliquer les signaux de la puce elle-même, avec une consommation électrique accrue [9].

U *"Une des approches possibles à cette issue est la séquentielle décomposent des opérations de telle manière qu'elle n'exige aucune variable supplémentaire autres*

> *que les variables disponibles comme entrée i.e., effectuer une transformation arbitraire d'objets par des transformations élémentaires locales successives à l'intérieur de ces objets et ceci sans considérer leurs états successifs. Comme ce paradigme de calcul in situ ne requiert aucune mémoire supplémentaire d'écriture, il trouve ses applications dans l'optimisation des programmes et la conception de puces électroniques en ce qui concerne le nombre de variables".*

Ces calculs permettent d'améliorer les performances de calculs et de calculer une opération liée à n registres par une séquence des affectations en utilisant seulement les n registres [63].

Nous commençons en s'appliquant la technique pour calculer une application d'un ensemble S^n vers lui-même par une séquence d'affectation. Nous pouvons considérer l'application $E : S^n \longrightarrow S^n$ comme le calcul parallèle de n affectation selon les n applications $S^n \longrightarrow S$ combinant l'application E, et ceci, soit en modifiant en même temps les n variables d'élément, ou exécuter l'application de n variables d'entrée sur n variables de sortie qui sont séparées.

Définition 1. *Si l'on veut calculer l'application E séquentielle-ment en modifiant les composants un par un et n'utilisant pas de mémoire autre que les variables d'entrée dont les valeurs finales écrasent les valeurs initiales, on doit nécessairement transformer les applications n de la forme $S^n \longrightarrow S$ d'une manière appropriée. Nous appelons "In Situ Design of Computation (IDC)" cette façon de calculer une application.*

Considérez l'exemple préliminaire suivant, celui est utile en comprenant l'idée fondamentale.

Exemple 1. *Considérons une application $E : \{0, 1\}^2 \longrightarrow \{0, 1\}^2$ définie par $E(x_1, x_2) = (x_2, x_1)$ décrivant un échange de deux variables booléennes. Un programme de base qui calcule E est : $x_0 := x_1$, $x_1 := x_2$, $x_2 := x_0$, alors que, une in situ programme $x_1 := f_1(x_1, x_2)$, $x_2 := f_2(x_1, x_2)$, $x_1 := g_1(x_1, x_2)$ for E évite l'utilisation de variables supplémentaires x_0, avec $f_1(x_1, x_2) = f_2(x_1, x_2) = g_1(x_1, x_2) = x_1 \oplus x_2$.*

Pour $S = \{0, 1\}$, l'application $S^n \longrightarrow S^n$, interprétant des réseaux booléens, ont été largement étudiés pour leur intérêt théorique en informatique et aussi pour leurs applications potentielles dans la nature (réseaux génétiques [37], réseaux de neurones, etc) ou dans les sciences sociales [34]. Les approches théoriques et combinatoires comme *in situ* calculs sont considérées comme précieuses dans le contexte de la conception de puces électronique en beaucoup de publications orientées électroniques, voir par exemple [58]. Du point de vue combinatoire, les assignations, qui sont des applications $S^n \longrightarrow S$, peuvent être considérée sous des formalismes différents. Par exemple, dans les réseaux d'interconnexion mufti-étages [1, 3, 74], une assignation est considérée comme un ensemble d'arêtes d'un graphe bipartite entre S^n et S^n où une pointe correspond à la modification de la composante concernée.
Les réseaux d'interconnexion mufti-étages (MIN) ont fait l'objet d'une recherche actif au cours des 40 dernières années. Beaucoup d'investigations existent au sujet de MINs examinant les applications diverses, par exemple [32, 76, 77, 82, 83, 84]. Les résultats au sujet des calculs *in situ* peuvent être traduits dans le contexte de MINs, parce que, faire des modifications successives de composants consécutifs de $X \in S^n$ est équivalent à l'acheminement d'un réseau de papillon (*i.e.* un hypercube bien ordonné) quand $S = \{0, 1\}$, ou le réseau de papillon généralisée a un plus grand degré pour un ensemble fini arbitraire S. Il a été prouvé que de telles *in situ* calculs existent pour la cartographie linéaire sur champ binaire et certains résultats combinatoires ont également été discutés [49].

1.1.1 Contexte d'optimisation

Au cours des dernières années, beaucoup d'informaticiens étudié le domaine de l'optimisation, y compris l'optimisation du compilateur, de l'optimisation du processeur, l'optimisation de code [4], l'optimisation de l'algorithme, etc. L'optimisation des problèmes spécifiques d'algèbre linéaire a été discutée à grande échelle parce qu'un tel type d'optimisations ont un effet significatif sur les performances du processeur. Whaley et Dongarra discutent optimiser les Basic Linear Algebra Subroutines (BLAS) dans [81]. Chatterjee et al. discutent des optimisations pour une suite de noyaux denses dans la matrice [15]. "Park et al. ont discuté l'intérêt d'utiliser la ré-allocation dynamique pour améliore la performance du cache dan la DFT dans [55]. Frigo, et al. in [23] ont discuté des performances du cache d'algorithmes de cache inconscients de la matrice transposée, FFT, et le tri. Optimisation des algorithmes bloqués a été largement étudié [38].

> *"une optimisation importante, affectant les performances du code du compilateur, est l'allocation des registres. Allocation de registres a été largement étudié dans la compilation et est un problème NP-complet. En 1981, Chaitin et al [13, 14] ont modélisé le problème de l'affectation de variables temporaires à la machine de k registres comme le problème de la coloration, avec k couleurs, le graphe d'interférence associé aux variables.".*

En général, l'accès de registre est plus rapide que l'accès de mémoire. Dorénavant il est préférable d'utiliser le registre que la mémoire chaque fois que c'est possible. Quand il n'est pas possible d'utiliser le registre alors quelques variables doivent être transférées à la mémoire. Cela charge/conserve l'opération a son prix. Pour éviter ce prix, quelques approches classiques ont été introduites comme dans les algorithmes de coloration de graphique [4]. Un algorithme itéré de coalescence de registres, proposé par Appel et George [24] est une version modifiée des développements précédents par Chaitin et al. [13, 14], et Briggs et al. [5]. Dans ces heuristiques, 'spilling', coalescence (enlever registre-à-registre mouvements), et coloration (assignant des variables aux registres) sont faites dans le même cadre. Ces techniques sont très utiles dans l'optimisation du compilateur, mais doivent encore être révisés pour obtenir de meilleurs résultats. Burckel et al. [8] ont introduit des méthodes de conception des programmes/circuits électroniques, pour effectuer toute opération sur k registres de toutes tailles dans un processeur, de telle manière que l'utilise sans mémoire de travail supplémentaire(tels que d'autres registres ou mémoires externes). De cette façon, toute opération est réalisée avec au plus $4k-3$ affectations de ces registres, ou $2k-1$ lorsque l'opération est linéaire ou bijective. Ces méthodes sont directement liés à l'optimisation du processeur et du compilateur.

Il est prouvé que toute les applications sur $\{0, 1\}^n$ peuvent être calculées par un programme *in situ* sans utiliser les variables supplémentaire et trois types des applications sont suffisants pour effectuer ces calculs [6, 10]. Toute les applications booléenne linéaire de dimension n peut être calculée avec une séquence de $2n-1$ d'affections linéaires [11, 12]. Chaque application E sur S^n peut être calculée par un *in situ* programme de longueur $5n-4$ et la borne a été améliorée à $4n-3$ lorsque $|S|$ est une puissance de 2. Par ailleurs, la longueur maximale du programme est $2n-1$, si les applications sont bijectives [8].

1.2 Édition collaborative décentralisée

La technologie mobile et la capacité de construire la connexion entre les appareils conduisent l'extension d'applications existantes dans de nouveaux royaumes, pour que les utilisateurs d'appareil mobiles puissent communiquer, traiter et présenter des informations. Édition collaborative

est une partie de travail de collaboration informatisé (Computer Supported Collaborative Work (CSCW)) [21], et est un important domaine de recherche en informatique depuis plus de deux décennies. Le développement rapide du réseau informatique a inspiré l'avancement de l'édition de collaboration en temps réel dans quels utilisateurs ne sont pas liés pour être dans le même endroit, ils peut être aux différents emplacements qui sont géographiquement dispersés. Ainsi, le système d'édition collaboratif peut être utilisée pour permettre aux gens, qui se trouvent à différents endroits, d'éditer un document textuelle partagé [20, 36, 60, **?**, 70, 71], de dessiner une structure de graphe partagé [25, 33], pour enregistrer les idées lors d'une réunion de discussion [27], ou de tenir une réunion de conception [50]. Ces documents pourraient être des articles, pages du wiki et du code source de programmation. Un éditeur collaboratif en temps réel peut être considéré comme une application logicielle partagée qui permet à plusieurs personnes de modifier un document partagé, à condition qu'ils soient reliés les uns aux autres par un système de réseau approprié.

Bien qu'ils soient des applications distribuées, les éditeurs collaboratifs temps réel (RCE) sont spécifiques au sens où ils doivent tenir compte des facteurs humains. Ainsi, elles sont caractérisées par les conditions suivantes [29] :

- Haute réactivité locale : le système doit être aussi réactif que son mono-utilisateur éditeurs [20, 70, 72] ;
- Une forte concurrence : les utilisateurs doivent être capables de modifier librement et en même temps n'importe quelle partie du document partagé et ceci à tout instant[20, 70] ;
- Cohérence : les utilisateurs doivent être capable de voir une vue convergeant de toutes les copies [20, 70] ;
- Coordination décentralisée : toutes les mises à jour concurrentes doivent être synchronisés de façon décentralisée afin d'éviter un point unique de défaillance ;
- Extensibilité : un groupe doit être dynamique dans le sens où les utilisateurs peuvent rejoindre ou quitter le groupe à tout moment.

Un éditeur de collaboration en temps réel peut être basé sur un système centralisé ou une architecture répliquée.

"Dans une architecture centralisée, un serveur central est responsable de garder le document partagé et de gérer les différents aspects de la collaboration, i.e., la cohérence, l'ordre des mises à jour, résoudre les conflits et l'adhésion à la session. Elle nécessite que chaque utilisateur propage son action vers le serveur central, puis le serveur appliquera ces modifications dans le document pour s'assurer de l'état du document destiné [17, 18] ".

"Dans une architecture décentralisée ou reproduite, il n y a pas de serveur central pour contenir le document partagé. Chaque site participant détient une copie du document partagé (réplique) sur lequel il travaille séparément. Chaque participant a besoin de diffuser ses actions / mises à jour de tous les sites participants afin que chaque site peut mettre à jour leurs répliques en conséquence. Par rapport à l'architecture centralisée, l'architecture décentralisée est plus attrayante, notamment en mode sans fil et en réseaux ad-hoc. L'absence de serveur central permet à l'utilisateur de poursuivre son travail même en cas de déconnexion. La collaboration est gérée par des dispositifs individuels en l'absence de serveur central [17, 18]".

Cette question pose des défis dans la mise en œuvre des éditeurs de collaboration dans une architecture reproduite, en particulier dans un réseau mobile qui se caractérise par la fiabilité

et la disponibilité de ressources limitées. Un des principaux défis techniques dans l'édition de collaboration en temps réel peut être expliqué dans l'exemple suivant.

Exemple 2. *Supposons que deux utilisateurs U_1 et U_2 démarrent avec un document partagé qui contient mot "Hear". l'utilisateur U_1 supprime le caractère 'H', puis insère le caractère 'N', afin que le mot est soit comme "Near". L'utilisateur U_2, avant qu'il ne reçoive les modification de U_1, supprime le caractère 'a' afin que la mot d'origine change en "Her". Les deux utilisateurs U_1 et U_2 reçoivent alors les mises à jour qui ont été appliquées aux versions du document qui n'a jamais existé sur leurs propres machines. Pourtant c'est un défi de comprendre comment appliquer les modifications des utilisateurs distants, qui ont été initialement créés dans des versions du document qui n'a jamais existé localement, et qui peuvent entrer en conflit la version éditée par l'utilisateur (expliqué plus en détail dans la partie II, Chapitre 7, la section 7.1.2).*

Par ailleurs, puisque les utilisateurs ont apporté des modifications contradictoires à leurs copies, ces copies multiples du même document peuvent causer une confusion. Les modifications conflictuelles doivent être intégrées dans un document cohérent. Lorsque les utilisateurs arrêtent de travailler sur différentes versions d'un document, *i.e*, suite à des retards de livraison, le problème de l'intégration devient encore plus complexe. Pour répondre à ces questions, l'indexation du contenu des documents partagés semble être la condition de base.

U *"une des approches possibles est que le document partagé soit mappé à un intervalle $I = [a, b]$ (où, $0 \leq a < b$ et $a, b \in \mathbb{R}$) de sorte que chaque ligne de caractère / ou d'objets contenus dans le document est associé à un identifiant unique dans l'intervalle I. Pour ce faire, nous introduisons une méthode d'indexation de précision de contrôle qui génère ces identifiants tels que les identifiants sont de type réels. Nous commençons dans le calcul de ces identifiants en utilisant la formule au milieu, avec modification particulière et les conditions d'arrondi. Il permet d'assurer que chaque utilisateur est capable d'insérer des éléments (caractères, lignes, etc) avec des identifiants différents dans le même intervalle et est capable de connaître la cardinalité locale et globale correspondant à l'ensemble des identifiants".*

1.2.1 Contexte de édition collaborative

Une comparaison de plusieurs approches au problème de l'édition d'un texte partagé en collaboration a été écrit par Ignat et al [28]. Un certain nombre d'applications collaboratives sont basées sur la transformation opérationnelle (OT) [20, 43, 44, 69, 70] mais OT considère des éditions collaboratives basées sur les opérations non commutatives et transforme les arguments des opérations réalisée à distance de telle sorte à prendre en compte les effets des exécutions simultanées. En fait, OT a besoin de deux conditions d'exactitude [59]. La transformation doit trepermet des opérations simultanées de sorte que l'exécution soit en ordre, et les fonctions de transformation eux-mêmes doivent suivre une commutation. La deuxième condition est plus complexe que la première [56]. Il est prouvé que toutes les transformations proposées précédemment ne garantissent pas ces propriétés [30, 31]. Par ailleurs, les solutions proposées plus tard [**?**, 52, 73] sont complexes, et leur exactitude ne peut être facilement vérifiée.

"Tous les algorithmes basés sur OT [20, 45, 51, 61, 70] appartiennent à l'approche de niveau logique. L'approche de niveau logique exige le maintient et l'analyse du journal historique pour décider des chemins de transformation pour les opérations de mise à jour effectuées à distance. La procédure de transformation devient couteuse en charges de travail. Par ailleurs, les opérations définies sur le point de vue logique perdent la position exacte des indices quand le document

est modifié, ce qui crée des difficultés pour annuler des opérations. OT-algorithme [51] introduit également "tomb stone" dans le cadre de leur structure de données. Mais le but est de résoudre les ambiguïtés sur les opérations de transformation qui mettent à jour la même portion d'un document.".

Oster et al. [54] a proposé l'algorithme WOOT pour gérer l'édition coopérative et pour assurer les opérations d'insertion de suppression. L'algorithme WOOT est également identifiée comme un CRDT (Commutative Replicated Data Types) [56]. Dans Woot, chaque caractère reçoit un identifiant unique, et maintient les identifiants des caractères précédents et suivants au moment de l'exécution initiale. Par ailleurs, la structure de données croît indéfiniment, pour la raison qu'il n'ya pas de 'garbage-collector' ou de restructuration [68].

L'algorithme WOOT [54] est la technique utilisée dans l'éditeur de TeNDaX [41] et appartient à l'approche de niveau physique. L'algorithme WOOT définit un site identifiant paire, compteur local pour créer un identifiant unique pour chaque caractère. Cette identifiant est unique pour chaque site. Le compteur local est incrémenté chaque fois qu'une nouvelle opération est exécutée. En TeNDax, chaque caractère se voit attribuer un identifiant unique intégrale par un serveur central. Dans les deux approches, les identifiants de caractères sont indexés (comme les tables de hachage ou B-arbres) afin qu'ils puissent être rapidement trouvé pour déterminer les dépendances de données entre les opérations. Le problème majeur dans leurs approches est que les identifiants ne transportent pas les informations de commande. Par conséquent, les caractères qui sont logiquement consécutives ne peuvent pas être stockées physiquement sur le disque de manière consécutifs [57].

R*" Récemment, Weiss et al. proposé le Logoot CRDT [79] où ils introduisent des identifiants comme des n-uplet $\langle i, s, c\rangle$ avec 'i' un chiffre, 's' un identifiant du peer 'c' une horloge. Une autre approche d'arbres binaires de Treedoc dense est proposé par Nuno et al. [56]. Weiss et al. introduit un identifiant de position comme une liste d'identifiants (longs) uniques. Par rapport à la technique Treedoc, Logoot a un surcoût élevé. Une autre technique pour assurer la cohérence est d'exécuter les opérations dans le même ordre pour toutes les copies (répliques) [40]. Pour s'assurer que la position d'édition a la même signification à toutes les répliques on a besoin, soit des répliques de fonctionnement en lock-étape, ou de la transformation opérationnelle [69]. Dans la conception Treedoc proposée par Nuno et al. [56], commune d'exécuter les opérations d'édition d'une manière optimale, ave une latence et synchronisation des répliques seulement en opérations de fond".*

Toutefois, nous avons remarqué que les deux algorithmes [56, 80] de génération d'identifiants uniques de position de de ligne, ne prend pas en charge certaines situations critiques. Nous discutons ces problèmes en citant des exemples construits en utilisant des algorithmes originaux.

1.2.2 Approche par Weiss et al.

Dans cette section, nous décrivons en détail l'algorithme pour générer les identifiants, proposée par Weiss et al [80]. L'algorithme (proposé par Weiss et al.) consiste en deux parties, Fonction et Fonction . Les auteurs envisagent une modification du document partagé comme un ensemble d'opérations, appelé patch. Donc, effectuer un ensemble d'opérations (patch) se traduit par l'insertion de lignes qui sont contigus ou regroupées par blocs contigus. Dans Fonction , N représente la taille d'un bloc. Ainsi, entre deux lignes indexés par deux identificateurs p et q de ligne, N

Fonction(Comment générer nouvelle ligne identifiant)

```
   Fonction : generateLineId
   Input : (p, q, N, boundary, site)
 1 begin
 2     let list := π
 3     index := 0
 4     interval := 0
 5     while (interval < N) do
 6         index++
 7         interval := prefix(q, index) − prefix(p, index) − 1
 8     end
 9     Let step := min(interval/N, boundary)
10     r := prefix(p, index)
11     for j :=1 to N do
12         list.add(constructId(r + Random(1, step), p, q, site))
13         r :=r + step
14     end
15     return list
16 end
```

désigne le nombre d'identifiants qui doivent être générées sur le site s. La variable 'boundary' est utilisée pour limiter la distance entre deux identificateurs de ligne successifs de telle sorte que les identifiants de ligne construit peuvent être réparties. Par conséquent, 'boundary' contribue à regrouper les identifiants de ligne afin que l'espace soit créé pour des insertion ultérieure. avec la valeur 'boundary' choisie, les identificateurs peuvent être répartis de façon aléatoire. On appelle ce processus : la stratégie aléatoire. La fonction $prefix(p, i)$ (voir ligne 10 , Fonction) renvoie un nombre en base BASE.

La deuxième partie de l'algorithme [80] est Fonction , comme indiqué ci-dessous, construit des identifiants et est exécuté grâce à la Fonction . En fait, la Fonction trouve une place pour les lignes N entre la plus courte possible, des préfixes d'identificateurs p et q. Après avoir sélectionné les préfixes les plus courts possible pour N, Fonction retourne les identifiants N.

Nous analysons les capacités des fonctions (et) dans le contexte du scenario d'échange des identifiants expliqué ci-dessous.

Scénario d'échange d'Identifiants

Par souci de simplicité, nous prenons deux utilisateurs U_1 et U_2 dans $site\,1$ et $site\,2$ (voir Figure 1.1) et un document vide avec des identifiants initiales et finales. Supposons que deux identifiants sont générés à $site\,1$ et $Site\,2$ respectivement correspondant à l'insertion de deux caractères "a" et "b". Supposons que les mises à jour sont exécutés et les identifiants sont échangés de telle sorte que chaque utilisateur voit deux identifiants. Notre analyse est que, si l'un des utilisateurs insert un nouveau caractère entre "a" et "b", un nouvel identifiant se trouve entre les identifiants correspondant aux caractères "a" et "b" ou pas ?

Fonction(Comment construire nouvel ID)

```
   Fonction : constructId
   Input : (r, p, q, site)
 1 begin
 2 |  let id :={}
 3 |  for i := 1 to |r| do
 4 |  |  d :=ith digit of r
 5 |  |  if d = p_i.digit then
 6 |  |  |  s := p_i.siteid
 7 |  |  |  c := p_i.clock
 8 |  |  |  else if d = q_i.digit then
 9 |  |  |  |  s := q_i.siteid
10 |  |  |  |  c := q_i.clock
11 |  |  |  |  else
12 |  |  |  |  |  s := site.identifier
13 |  |  |  |  |  c := site.clock++
14 |  |  |  |  end
15 |  |  |  end
16 |  |  end
17 |  |  id := id.⟨d, s, c⟩
18 |  |  return id
19 |  end
20 end
```

Observons "Approche par Weiss et al." dans le scénario d'échange d'identifiant. Selon cette approche, pour un document vide, nous prenons les identifiants initiales et finales $\langle 0,\ NA,\ NA\rangle$ et $\langle MAX,\ NA,\ NA\rangle$ respectivement, où $MAX = BASE-1$. Après avoir échangé des identifiants entre deux utilisateurs, chaqu'un peut avoir quelques uns des identifiants sous la forme $\langle i,\ s_j,\ c_j\rangle$, pour $j = 1,\ 2, \ldots$, et $i = 0,\ 1, \ldots, MAX$. Supposons que ces deux identifiants sont

$$\langle i,\ s_1,\ c_1\rangle \text{ et } \langle i,\ s_2,\ c_2\rangle$$

$$s_1 < s_2 \text{ or } s_1 = s_2 \text{ et } c_1 < c_2$$

Supposons que l'utilisateur U_1 a l'intention d'insérer un nouveau caractère entre les deux groupes qui sont déjà là. Nous détectons certains inconvénients dans l'algorithme (constituée des fonctions et) qui sont donnés ci-dessous.

Itération Infini : Pour deux identifiants qui ont les mêmes valeurs de i, Algorithme (composé de la fonction Fonction et Fonction) ne génère pas de nouvel identifiant pour la raison que la valeur "$interval = -1$" (voire ligne 7, Fonction). Il exécute la boucle pour une itération infinie.

Supposons qu'on quitte maintenant l'itération infinie en définissant des conditions supplémentaires pour l'algorithme, on aura toujours un problème pour préserver l'ordre comme expliqué ci-dessous.

L'altération d'ordre : Pour insérer un identificateur, nous pouvons prendre la valeur $N = 1$. Puisque la sélection de valeur pour 'boundary' est optionnel, on peut donc prendre

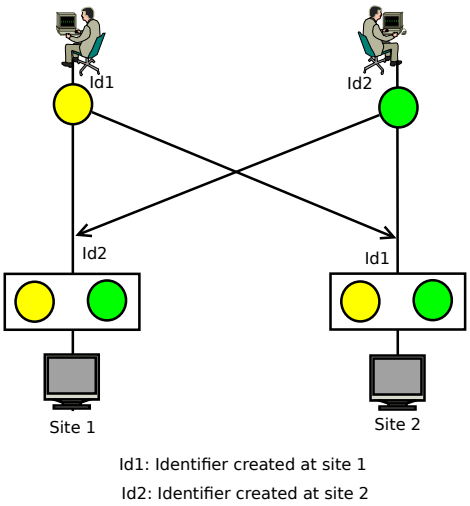


FIGURE 1.1 – Scénario d'échange d'Identifiants

"$boundary = 1$" puis pour deux identifiants $p = \langle 0,\ s,\ c\rangle$ et $q = \langle 0,\ s,\ c\rangle$ avec $p < q$ (say), L'algorithme rend des valeurs comme donné ci-dessous

$$\text{step} = min(1,\ 1) = 1 \ \text{ (see line 9, Function)}$$

et

$$\text{prefix}(p,\ index) = 0 \Longrightarrow \text{r} = 0 \ \text{ (see line 10, Fonction)}$$

Après l'exécution de la Fonction pour ces valeurs, la Fonction retourne weissf2 nouvel identifiant comme

$$id = \langle 1,\ s,\ c\rangle \ \text{ (see line 17, Fonction)}$$

Pour préserver notre ordre désiré, les identificateurs calculés devraient suivre

$$p = \langle 0,\ s,\ c\rangle < \langle 1,\ s,\ c\rangle < \langle 0,\ s,\ c\rangle = q$$

qui n'est pas vrai parce que $1 \notin [0,\ 0]$.

Échec de formule : La formule proposée (par Weiss et al.) pour calculer le nombre d'identificateurs entre deux identificateurs ne supporte pas certain cas. Par exemple, supposons que deux lignes sont identifiées par les identificateurs

$$p = \langle k_1,\ 4,\ 7\rangle\langle k_2,\ 9,\ 5\rangle \quad \text{et} \quad q = \langle k_3,\ 5,\ 3\rangle\langle k_4,\ 3,\ 6\rangle\langle k_5,\ 3,\ 9\rangle$$

avec l'hypothèse que $BASE = 100$, $boundary = 10$ et

$$\text{Supposons que } k_1 = k_2 = k_3 = k_4 = k_5 = k$$

Supposons que nous nous intéressons à insérer des lignes de N entre deux lignes identifiées par les identificateurs p et le q, ensuite calculer le nombre d'identifiants en utilisant la formule comme suit

$$\text{prefix}(q,\ 1) - \text{prefix}(p,\ 1) - 1 = \begin{cases} -1 & \text{if } k_3 = k_1 \\ 0 & \text{if } k_3 = k_1 + 1 \end{cases}$$

Ainsi, la formule ne fonctionne pas dans ce cas, d'ailleurs, ça ne fonctionne pas pour calculer la cardinalité des identifiants pour les tailles de 2, 3 et ainsi de suite, pour les valeurs prescrites ci-dessus.

Redondance des Identificateurs : Nous expliquons la redondance des identifiants en considérant le scénario suivant. Supposons que deux utilisateurs sur deux sites différents *site* 1 et *site* 2 ont un document partagé vide (voire Figure 1.2) avec le point initial et final $lb =< 0, NA, NA >$ et $le =< 10, NA, NA >$ respectivement. Maintenant l'utilisateur U_1 (avec $s = 1$ et horloge initiale $c = 0$)

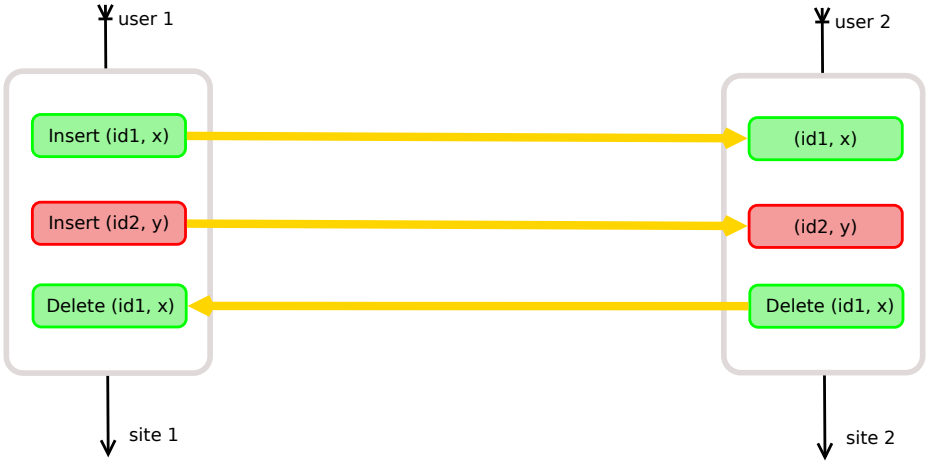


FIGURE 1.2 – Redondance des Identificateurs

1. Insert (id_1, x), insère caractère "x" avec un identifiant "id_1".
2. Delete (id_1, x), supprime caractère "x" avec un identifiant "id_1".
3. Insert (id_2, y), insère caractère "y" avec un identifiant "id_2".

insère un caractère "x" (*i.e.* $N = 1$) et crée l'identifiant p. L'algorithme (Fonction , Fonction), prend la valeur $p =< 9, 1, 1 >$ (voir ligne 4 à ligne 17, Fonction). Encore une fois, supposons que l'utilisateur U_1 insère un autre caractère "y" et crée un identifiant q entre $p =< 9, 1, 1 >$ et $le =< 10, NA, NA >$. L'algorithme (Fonction , Fonction) génère un nouvel identifiant q qui pourrait être égale à p ou le. Soit $q = p$, mais puisque le contenu de p et q ne sont pas nécessairement les mêmes. il faut permettre au contenu de p d'être "x" et le contenu de q d'être "y". Les mises à jour sont envoyées à l'utilisateur U_2 au *Site* 2. Maintenant supposons que l'utilisateur U_2 a l'intention d'effectuer l'opération "Delete (id_1, x)" puis envoie la mise à jour à l'utilisateur U_1. L'exécution de " Delete" au *Site* 2 (à cause des identificateurs redondant) ne garantit pas le résultat souhaité et peut mène à une divergence.

1.2.3 Approche par Preguiça et al.

Nuno preguiça et al. [56] proposent une approche pour construire des identifiants de position unique dans lequel ils introduisent des disambiguators uniques. Ils ont appelé cette approche $UDIS$. L'algorithme proposé est (Fonction) décrit ci-dessous. Nous analysons Fonction aussi le Scénario d'échange des identificateurs décrit dans la section 1.2.2 (Figure 1.1).

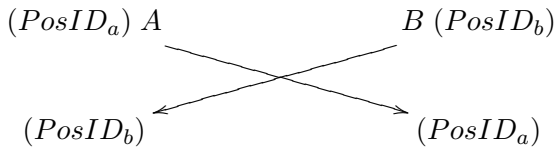
L'altération d'ordre Supposons qu'un document vide est marqué avec l'identificateur initial et final $(0, d_0)$, et $(1, d_1)$ respectivement. Supposons que deux utilisateurs A et B éditent et

Fonction(Générant POSID)

Result : newPosID ($PosID_p$, $PosID_f$)

```
1  // d : new disambiguator.;
2  Require : PosID_p < PosID_f ∧ ∄ atom x such that PosID_p < PosID_x < PosID_f;
3  begin
4  |  if PosID_p/+PosID_f then
5  |  |  Let PosID_f = c_1 ⊙ ... ⊙ (p_n : u_n);
6  |  end
7  |  return c_1 ⊙ ... ⊙ p_n ⊙ (0 : d);
8  |  else if PosID_f/+PosID_p then
9  |  |  Let PosID_p = c_1 ⊙ ... ⊙ (p_n : u_n);
10 |  end
11 |  return c_1 ⊙ ... ⊙ p_n ⊙ (1 : d);
12 |  else if MiniSibling(PosID_p, PosID_f ∨ ∃PosID_m > PosID_p
13 |  : MiniSibling(PosID_p, PosID_m ∧ PosID_m/+PosID_f then
14 |  |  Let PosID_p = c_1 ⊙ ... ⊙ (p_n : u_n);
15 |  end
16 |  return PosID_p ⊙ (1 : d)
17 end
```

modifient le document et s'échange les identifiants correspondant (par exemple, $PosID_a$, $PosID_b$) en exécutant des mises à jour.



Après avoir effectué les mises à jour et l'échange d'identificateurs, chaque utilisateur verra deux identifiants. Maintenant, supposons que l'utilisateur B a l'intention de créer un nouvel identifiant entre deux identifiants (déjà calculé, l'un d'eux est envoyé par un autre utilisateur). Ce nouvel identifiant est calculé par la fonction . Le problème est expliqué en détails sur la figure 1.3. Figure 1.3 présente un document vide avec le point initial et final marqué comme 'Beg' et 'End' respectivement. Maintenant n'importe quel nouvel identifiant de position doit être le fils droit du nœud 'Beg' ou le fils gauche du nœud 'End'. L'utilisateur A veut insérer un caractère "a" et l'utilisateur B veut insérer un caractère "b" à la même position. Soit dA et dB les disambiguators pour les utilisateurs A et B tel que $dA < dB$ alors, selon le premier "if condition" (voir les lignes du 4 à 6) dans Fonction

$$PosID_a = [1(0 : dA)] \text{ et } PosID_b = [1(0 : dB)]$$

Maintenant deux nœuds marqués comme "a" et "b" sont des frères. Supposons que l'utilisateur B veut insérer un nouveau caractère "a_1" entre "a" et "b" alors il faut qu'il crée un nouvel identifiant selon "if condition" (voir lignes 12 à 15) de Fonction afin que

$$PosID_{a_1} = [1(0 : dA)(1 : dB)]$$

Soit

$$dA = (counter,\ siteID) = (1,\ 1)$$

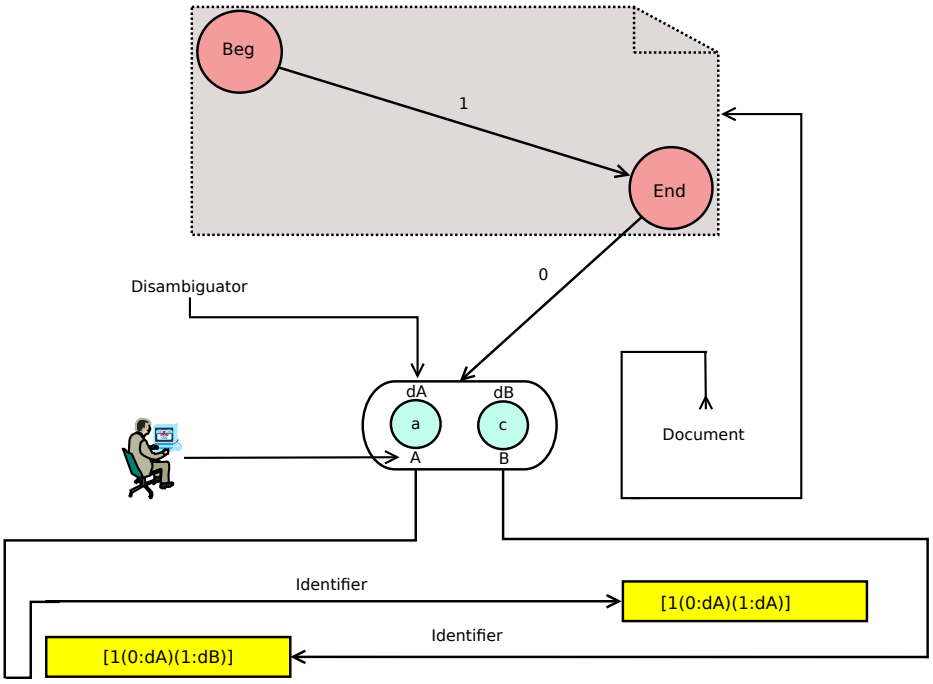


FIGURE 1.3 – Générant POSID

et

$$dB = (counter,\ siteID) = (1,\ 2)$$

puis

$$PosID_a = [1(0 : (1,\ 1))]$$

et

$$PosID_b = [1(0 : (1,\ 2))]$$

Maintenant, identifiant pour le caractère "a_1" est calculé comme

$$PosID_{a_1} = [1(0 : (1,\ 1))(1 : (1,\ 2))]$$

Suite à la définition tel que prescrit par Nuno preguiça et al. [56], il donne

$$PosID_a < PosID_{a_1} \text{ But } PosID_b < PosID_{a_1}$$

$$i.e.,\ PosID_{a_1} \notin [PosID_a,\ PosID_b]$$

Pour faire face à ce genre de problèmes que posent la restriction pour les utilisateurs, nous avons l'intention de fournir un environnement où les identifiants suivent un ordre strict et peuvent être vu comme un ensemble ordonné. Ceci offrira une option pour les participants de démarrer, alors que, l'ensemble des identifiants sera un ensemble ordonné fini qui préserve un ordre strict.

1.3 Contributions

Nous concevons des algorithmes pour calculer les applications d'un ensemble S^n vers lui-même par séquence d'affectation linéaires. Ce calcul séquentiel ne nécessite aucune variable

supplémentaire autre que les variables disponibles dans la saisi. Il peut aussi être interprété comme étant l'exécution d'une transformation arbitraire d'objets par des transformations locales élémentaires successives à l'intérieur de ces objets et seulement en ce qui concerne leurs états successifs. Ce *"In Situ Design of Computation (IDC)"* n'exige pas de mémoire d'écriture supplémentaire, par conséquent, il trouve des applications dans l'optimisation des programmes et la conception de circuits. Ce *" calcul in situ"* permet d'améliorer les performances de calculs et de calculer une opération liée aux registres n par une séquence de tâches en utilisant uniquement ces n registres.

"Nous étudions, vérifions et prouvons l'existence d'un tel "Conception in situ de calcul" sur les corps, les anneaux et les entiers. Par ailleurs, nous étendons l'étude au cas des booléen afin de traiter les polynômes booléens et élargir l'idée à d'autres domaines de recherche. Nous mettons en œuvre "In Situ Design of Computation (IDC)" par la conception d'algorithmes et le calcul de tels applications avec succès. Tous les programmes in situ considérés au cours de ces travaux opèrent sur les composantes consécutives, en traversant la liste de tous les indices avec la possibilité de traverser plusieurs fois la liste dans l'ordre avant ou arrière".

`En contribuant au système d'édition collaborative.`

Nous visons à concevoir un système d'édition de collaboration décentralisé (DCES) et étudier ces limitations existantes. Nous visons à ce que ce le système soit capable à traiter les applications modernes qui sont basées sur les techniques d'édition de collaboration utilisées récemment. Certaines des limites (*par exemple*, nécessitent un indexage correcte des caractère/lignes ou objets compris dans un document partagé. Pour l'indexation appropriée, il doit créer des identifiants pour chaque caractère/ligne ou objet. Quelques techniques sont disponibles pour produire ces identifiants, par exemple [56, 80]. Ces techniques évitent d'employer des nombres réel en raison des limites de précision et les algorithmes proposés ne préservent pas l'ordre de ces identifiants, particulièrement pendant les calculs réalisés à distance et aussi pendant les échanges des points.

"Nous introduisons une méthode d'indexation de précision de contrôle pour générer un ensemble d'identifiants uniques tels que ces identifiants soit utilisés pour l'indexation des caractères/lignes ou des objets. Ces identifiants sont toujours des nombres réels avec un modèle de précision spécifique et contrôlé. L'ensemble de ces identifiants est finie et permet de calculer des cardinalités locales et globales. Pour générer un nouvel identifiant, seules les informations sur deux identifiants voisins est nécessaire. A partir d'un identifiant, le site, où il a été généré, peut être identifié immédiatement. Nous commençons par indexer le document partagé, initialement, avec un intervalle $I = [a, b]$ avec $0 \leq a < b$ for $a, b \in \mathbb{R}$ et de nouveaux identifiants se trouvent dans l'intervalle I. Comme système d'édition de collaboration nous utilisons un réseau $\aleph$ de n ($n \in \mathbb{N}$) utilisateurs/sites or peers tels que chaque site/peer se voit assigné un identifiant produit avec la précision spécifiée. Nous mettons en œuvre avec succès l'idée en concevant un algorithme qui préserve l'ordre de l'ensemble des identifiants de l'ensemble et ces sous-ensembles".

1.4 Vue d'ensemble et structure

Une vue d'ensemble de cette thèse est exposée ci-dessous.

La partie I discute et développe des stratégies pour la défaillance séquentielle d'opérations.

- ⋆ La chapitre 2, donne un aperçu général rapide de l'optimisation de compilateur/processeur et fournit des motivations.
- ⋆ Chapitre 3, commence dans fournir le concept fondamental de *"In Situ Design of Computation (IDC)"* pour les applications, explique son existence sur les corps, donne la stratégie alternative et illustre avec exemples.
- ⋆ Chapitre 4, verifies l'existence de *"IDC"* pour les applications sur les anneaux, explique et discute le cas des applications linéaires inverses.
- ⋆ Chapitre 5, développe de existence de *"IDC"* pour applications sur nombres entiers, conçoit des relations pour déterminer la limite sur le nombre des affectations.
- ⋆ Chapitre 6, discute *"IDC"* pour les applications booléennes bijective et générales et fournit la fondation pour développer *"IDC"* par les polynômes sur GF(2).

Partie II présente, le système d'édition collaborative décentralisé, basée sur la méthode d'indexage de précision contrôlé.

- ⋆ Chapitre 7, présente le matériel d'introduction sur les systèmes d'édition collaborative décentralisée et définit le modèle d'édition collaborative.
- ⋆ Chapitre 8, introduit stratégie d'indexation de précision contrôlé, illustre la façon de générer des identifiants uniques et fournit la méthode pour calculer la cardinalité.
- ⋆ Chapitre 9, décrit les propriétés d'identifiants uniques, analyser la méthode, pour générer des identifiants, sous certaines conditions.
- ⋆ Chapitre 10, fournit des modèles différents, pour l'échange de points, afin d'observer la variation de cardinalité d'identifiants uniques.

Partie III Présenté une evaluation.

- ⋆ Chapitre 11, décrit le travaux de future et présente conclusion.

Première partie

Décomposition séquentielle des opérations

Chapitre 2

L'optimisation de compilateur/processeur (Un aperçu rapide)

2.1 Motivations

Processeurs multi-cœur apportent une évolution de la technologie informatique. Presque, tous les ordinateurs modernes sont équipés de processeurs multi-cœur. Processeurs multi-cœur offrent des avantages de performance et de productivité au-delà des capacités de single-cœur aujourd'hui les processeurs. Cette nouvelle tendance des processeurs multi-cœur exigeants nouvelles modifications dans les compilateurs afin qu'ils puissent être en mesure de faire face à de nouveaux défis. Compilateurs modernes sont introduites avec un nombre significatif d'optimisation qui ont des effets différents sur la qualité et la taille du code, le temps écoulé et la consommation d'énergie etc Le code du programme qui doit être compilé influencé l'optimisation aussi bien et l'optimisation du code introduit la nouvelle tendance à la fois dans les hardwares et softwares [26, 64, 65].

Les architectures actuelles contiennent un ou plusieurs processeurs qui sont équipés d'un nombre relativement restreint de registres. Le nombre de registres disponibles sur un processeur et les opérations qui peuvent être effectuées en utilisant ces registres ont un impact significatif sur la qualité du code généré par l'optimisation des compilateurs. Ces registres sont constamment demandés dans les opérations, cependant, il est conseillé de minimiser l'utilisation de registres. Il est prouvé qu'il est possible de calculer une opération liée aux registres n par une séquence de tâches en utilisant uniquement ces n registres. Par ailleurs, si cette opération est linéaire ou bijective, le nombre d'affectations est au plus $2n$. Dans le cas général, ce nombre de missions est au plus $4n$.

Il est un fait que les architectures informatiques actuelles atteignent leurs limites théoriques de la performance. Cependant, il est encore possible de gain de performances de calcul en effectuant des calculs d'une manière nouvelle. Ce type de calculs, en fait, généraliser le principe traditionnel de l'échange de deux nombres A, B par la séquence :

$$A := A + B$$
$$B := A - B$$
$$A := A - B$$

Nous visons à étudier ces méthodes et foyer pour améliorer ces techniques des calculs, en particulier pour développer des méthodes et des algorithmes heuristiques efficaces pour trouver ces décompositions et mettre en œuvre ces méthodes. Nous sommes intéressés par l'amélioration des limites sur ces calculs, pour obtenir de nouvelles méthodes de calculs pour des cas particuliers et mettre en œuvre ces méthodes dans un compilateur avec un objectif de l'optimisation du code en langage machine. Cette mise en œuvre peut être fournie dans le matériel à travers la conception de nouveaux processeurs, dans le logiciel, par les compilateurs optimisant en amont (compilateurs de langages pré-niveau) et en aval (compilateurs de langages de post-machine).

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 3

Conception *in situ* de calcul pour les applications linéaires sur des corps

Calcul séquentiel des applications linéaires de telle façon qu'il ne nécessite aucun variables supplémentaires autres que les variables disponibles en entrée, est une approche vers l'optimisation (par exemple, le processeur, du compilateur, la mémoire, le code). Une étape fondamentale dans cette approche est d'enquêter sur l'existence de tels calculs sur les champs, *i.e*, si les coefficients de missions linéaires (impliquer dans le calcul de la cartographie donnée) appartiennent à un domaine ou non.

3.1 Calcul séquentiel

La conversion de l'entrée en une sortie selon une séquence bien définie d'étapes de calcul conduit vers la théorie du calcul séquentiel. Un programme séquentiel met en application la fonction mathématique qui trace un ensemble d'entrées à un ensemble de sorties. Ces fonctions mathématiques sont bien définies et la notion des fonctions calculables a été présentée plus tôt par Church, Kleene et Turing. Ces fonctions sont fréquemment utilisées dans le untyped lambda calculus, fonctions récursives, et les machines de Turing [16, 35, 75].
Ces modèles de base aident en concevant et le raisonnement pour des langages de programmation, théorie de domaine et sémantique de dénotationnel présentée par Scott et Strachey, et fournissent un arrangement mathématique global pour le calcul séquentiel, constructent sur les théories fondamentales [67]. Ce progrès, interconnecte les langages de programmation différents et permet une connexion avec le monde mathématique de l'algèbre, la topologie et la logique. Il inspire les langages de programmation, les disciplines de type, et les méthodes de raisonnement.

"Un in-place algorithme convertit structure de données en utilisant un minimum d'espace de stockage supplémentaire constant. Lorsque ces algorithmes courir, l'entrée est écrasée par la sortie. Par exemple, trier tas est un algorithme (in-place) de tri ".

" Une opération est dite une in-place l'opération si elle ne modifie pas l'état normal du système, comme une sauvegarde de fichiers peuvent être stockés sur un système exécutant sans altérer la vitesse du système, tout en en place une opération dépend de la sophistication du système".

Afin d'améliorer les performances du cache, un algorithme ou d'une application devrait augmenter la réutilisation des données, la diminution des conflits de cache, et la pollution du cache diminuer, car une grande quantité de pollution du cache augmente le besoin en bande passante de la demande, même si l'application n'est pas en utilisant des données plus [48].

La technique, nous utilisons, est basé sur le principe de la réutilisation, seule, l'ensemble des variables d'entrées disponibles dans le calcul de la cartographie séquentielle.

3.1.1 *In Situ* Design of Computation (IDC)

Dans cette section, nous expliquons le concept de *"In Situ Design of Computation (IDC)"* et interpréter un simple application linéaire.

Burckel et al. (voir par exemple [9]) introduisent une technique pour calculer les applications par séquence de effectuations linéaires, qui sont toujours applications linéaires. Cette suite d'applications linéaires réutilise l'ensemble des variables d'entrée disponibles et ne pas utiliser n'importe quelle variable supplémentaire. Cette méthode de calcul est connu comme *"In Situ Design of Computation (IDC)"* des applications. Un effet ultime de cette technique entraîne décomposition séquentielle d'opérations.

Elle commence dans la construction de la première affectation de la matrice qui représente les coefficients de la cartographie donnée, puis calculer sa valeur de référence que l'utilisation dans les prochaines missions où jamais elle avait besoin pour être référencé. Chaque affectation linéaire, consiste dans la séquence, est limitée à avoir un nombre maximum de variables, pas plus que disponibles dans l'ensemble initial de variables d'entrée. Chaque mission est construite de telle manière que sa référence peut être calculée, et la référence calculé est utilisé dans les affectations prochaines impliquer dans la séquence, si elle est nécessaire là-bas.

Interpréter une affectation linéaires

Pour interpréter une affectation linéaire, un diagramme est montré dans Figure 3.1. Supposons que 'SOIV' désigne l'ensemble des variables d'entrée et 'v' désigne une variable *ith* de l'ensemble des variables d'entrée (SOIV). Delta contient des éléments de l'ensemble 'SOIV' mais il ne dépend pas de la *ith* variable 'v', et 'alpha' (agit comme coefficient pour une affectation) désigne un élément de champ (disons par exemple) et il doit être non nulle pour une affectation soit inversible. Une affectation linéaire consiste à ces paramètres, est présenté par barre verte dans la Figure 3.1. Un petit écart dans le bloc blanc brun foncé rend différence entre 'SOIV' et 'Delta'.

Comment calculer la valeur de référence d'une affectation unique impliquant dans la séquence des tâches, est interprété dans la Figure 3.2. Dans cette Figure 3.2, 'beta' désigne la valeur inverse du 'alpha' qui se multiplient avec des blocs de couleur rose, d'affectations, où, 'v' est la valeur actualisée de la *ith* variable qui pourrait être utilisé comme valeur de référence , si elle est nécessaire, dans les prochaines missions impliquent dans la séquence de tâches qui calculent une cartographie donnée.

À titre d'exemple, supposons que

$$\text{SOIV} = \{x_1, x_2, x_3, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n\}$$

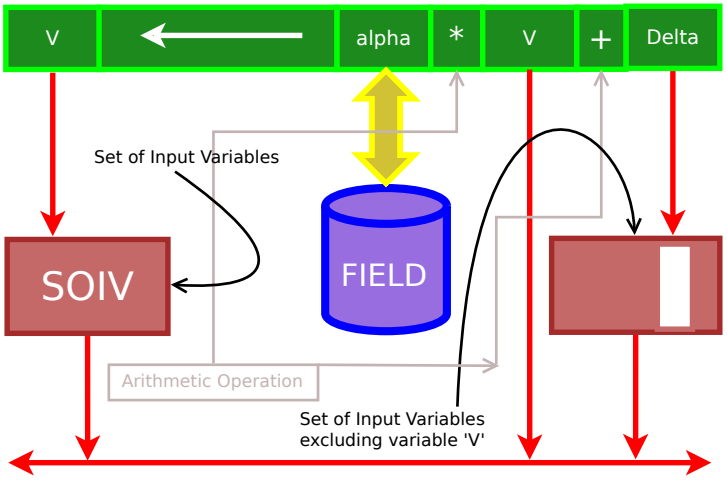


FIGURE 3.1 – Interpréter une affectation linéaires

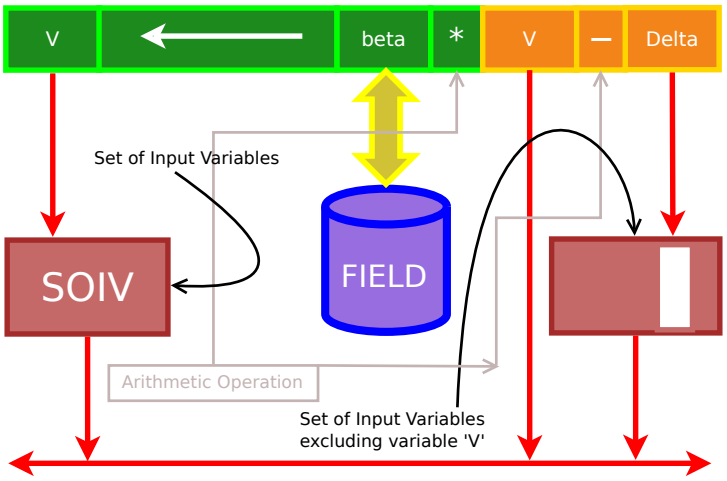


FIGURE 3.2 – Valeur de référence de l'affectation linéaire.

Pour un *ith* variable $x_i \in$ SOIV, une affectation est

$$x_i := alpha(x_i) + Delta$$

où,

$$Delta = \{x_1, x_2, x_3, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$$

La valeur de référence est calculé comme

$$x_i := beta(x_i - Delta)$$

où,

$$beta = alpha^{-1} \neq 0$$

3.2 Existence d'IDC pour les applications linéaires sur des corps

Dans cette section, nous expliquons que le calcul séquentiel des applications par séquence de affectations linéaires (Conception *in situ* de calcul), tels que les coefficients de ces affectations linéaires appartiennent au Corps, existent. Nous expliquons l'affirmation (voir [7]) par l'exemple suivant.

Exemple 3. *Considérons une application linéaire*

$$E\,(x_1,\, x_2,\, x_3) \longrightarrow (5x_2 + 7x_3,\, x_2 + 4x_3,\, 3x_1 + 2x_3)$$

L'application E peut être écrite comme $X := AX$ pour un vecteur $X := (x_1, x_2, x_3)$ et une matrice A de coefficients, étant donné que sous

$$A = \begin{pmatrix} 0 & 5 & 7 \\ 0 & 1 & 4 \\ 3 & 0 & 2 \end{pmatrix}$$

Parce que $a_{11} = 0 = \alpha$. Mais $a_{13} = 3 = \beta$. Par conséquent, nous devons soustraire la troisième rangée de la matrice A de la première rangée. L'opération sera $R_1 := R_1 - R_3$.
Après l'achèvement de cette opération, la matrice A prendra la forme :

$$A = \begin{pmatrix} -3 & 5 & 5 \\ 0 & 1 & 4 \\ 3 & 0 & 2 \end{pmatrix}$$

Maintenant, la première affectation sera de la forme :

$$x_1 := -3x_1 + 5x_2 + 5x_3 \tag{3.1}$$

La valeur initiale de x_1 sera

$$\frac{1}{3}(-x_1 + 5x_2 + 5x_3)$$

Maintenant, nous allons effectuer l'opération

$$x_2 := x_2 - \beta x_1 + \beta \left\{\alpha^{-1}\,(x_1 - \Delta)\right\}$$

et

$$x_3 := x_3 - \beta x_1 + \beta \left\{\alpha^{-1}\,(x_1 - \Delta)\right\}$$

La matrice A prendra la forme

$$A = \begin{pmatrix} -3 & 5 & 5 \\ 0 & 1 & 4 \\ -1 & 5 & 7 \end{pmatrix}$$

La deuxième affectation sera

$$x_2 := x_2 + 4x_3 \tag{3.2}$$

Maintenant, la valeur initiale du deuxième affectation, x_2 sera

$$x_2 - 4x_3$$

et pour le calcul de la troisième affectation :
Pour, $\alpha = 1$ et $\beta = 5$, Nous effectuons les opérations suivantes :

$$x_3 := x_3 - \beta x_2 + \beta \left\{ \alpha^{-1} \left(x_2 - \Delta \right) \right\}$$

Nous obtenons la troisième affectation de

$$x_3 := -x_1 + 5x_2 - 13x_3 \tag{3.3}$$

La première affectation de la deuxième séquence qui est obtenue par l'opération $R_1 := R_1 - R_3$ sera.

$$x_1 := x_1 + x_3$$

Ainsi la séquence requise des affectation est

$$\textit{Sequence of assignments} \rightarrow \begin{cases} x_1 := -3x_1 + 5x_2 + 5x_3 \\ x_2 := x_2 + 4x_3 \\ x_3 := -x_1 + 5x_2 - 13x_3 \\ x_1 := x_1 + x_3 \end{cases}$$

3.2.1 Calculer les applications inverse

L'application inverse est calculée en inversant les affectations et la réécriture de bas en haut.

$$\textit{Sequence of assignments} \rightarrow \begin{cases} x_1 := x_1 - x_3 \\ x_3 := \dfrac{1}{13} \left(-x_1 + 5x_2 - x_3 \right) \\ x_2 := \dfrac{4}{13} x_1 - \dfrac{7}{13} x_2 \\ x_1 := \dfrac{1}{3} \left(-x_1 + 5x_2 + 5x_3 \right) \end{cases}$$

3.3 Une approche utilisant l'identité de Bézout

Dans cette section, nous présentons une approche alternative (qui utilisent Bézout's Identité) pour construire une séquence d'affectation linéaire qui calcule application linéaire donnée. Nous le prouvons, ici, pour deux dimensions applications linéaires.

Définition 2. *Déclare l'identité de Bézout's que, si deux entiers a et b sont premiers entre eux alors il existe u, $v \in \mathbb{Z}$ tels que $au + bv = 1$.*

Exemple 4. *Soit E une application linéaire telle que définie ci-dessous*

$$E : (x, y) \longrightarrow (55x + 89y, 34x + 21y)$$

tels que la matrice A représente les coefficients.

$$A = \begin{pmatrix} m & n \\ p & q \end{pmatrix} = \begin{pmatrix} 55 & 89 \\ 34 & 21 \end{pmatrix}$$

Les valeurs correspondant aux entrées m et p sont co-prime, i.e., $GCD(55, 34) = 1$. Par conséquent, nous appliquons l'identité de Bézout pour trouver des entiers u et v tel que $55u + 89v = 1$

Un certain nombre de valeurs sont possibles pour les u et v. Deux d'entre eux sont $u = 13$, $v = -21$.
Multipliez la première rangée de la matrice A par u et la deuxième par v, il donne

$$\acute{A} = \begin{pmatrix} um & un \\ vp & vq \end{pmatrix} = \begin{pmatrix} 715 & 1157 \\ -714 & -441 \end{pmatrix}$$

Utilisent

$$a = -\frac{fp-m}{e},\ b = -\frac{fq-n}{e},\ c = -\frac{pe}{fp-m},\ d = -\frac{mq-np}{fp-m} \tag{3.4}$$

Nous trouvons les valeurs $a = 1$, $b = 716$, $c = -714$, $d = 510783$, $e = 1$, $f = -1$
Par conséquent, le séquence des affectations est, comme indiqué ci-dessous :

$$\begin{cases} x := x + 716y \\ y := \dfrac{1}{13}\left(-714x + 510783y\right) \\ x := -\dfrac{1}{21}\left(x - y\right) \end{cases}$$

3.3.1 Calculer les applications inverse :

Dans cette section, nous expliquons par exemple, comment calculer les applications inverse.

Exemple 5. *La séquence intermédiaire des affectations qui calculent l'application E, dans exemple 4, dont les coefficients sont représentés par la matrice*

$$\acute{A} = \begin{pmatrix} 715 & 1157 \\ -714 & -441 \end{pmatrix}$$

est donné ci-dessous

$$\begin{cases} x := x + 716y \\ y := -714x + 510783y \\ x := x - y \end{cases} \tag{SOA}$$

Nous construisons la séquence de affectations qui calculent l'application inverse, en inversant et la réécriture de la séquence de affectations SOA, comme ci-dessous.

$$\begin{aligned} x &:= x + y \\ y &:= \frac{1}{510783}\left\{714x + y\right\} \\ x &:= x - 716y \end{aligned}$$

et la matrice inverse $\acute{A}^{-1}$ est obtenue en évaluant la séquence de affectations qui calculent l'application inverse.

$$\acute{A}^{-1} = \begin{pmatrix} -\dfrac{21}{24323} & -\dfrac{89}{39291} \\ \dfrac{34}{24323} & \dfrac{55}{39291} \end{pmatrix}$$

Nous calculons A^{-1} en faisant quelques changements dans la séquence ci-dessus des affectations, i.e., nous multiplions la première colonne de $\acute{A}^{-1}$ par u et deuxième v. La séquence de affectations

qui calculent A^{-1} est donné ci-dessous :

$$\begin{aligned} x &:= u * x + v * y \\ y &:= \frac{1}{510783}\{714x + v * y\} \\ x &:= x - 716y \end{aligned}$$

La matrice A^{-1} correspondant à la matrice A est donné ci-dessous :

$$A^{-1} = \begin{pmatrix} -\frac{21}{1871} & \frac{89}{1871} \\ \frac{34}{1871} & -\frac{55}{1871} \end{pmatrix}$$

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 4

Conception *in situ* de Calcul pour les applications linéaire sur les Anneaux

Étudier l'existence de *"Conception in situ de calcul"* pour applications linéaires sur des anneaux est constitué, en fait, à trouver si les coefficients de ces *"Conception in situ de calcul"* appartiennent à des anneaux ou non. L'existence de Conception *in situ* de calcul, des applications linéaires sur des anneaux est prouvé dans [9]. Nous sommes intéressés à vérifier et à mettre en œuvre l'existence de *"In Situ Design of Computation (IDC)"* sur des anneaux.

Dans ce chapitre, nous expliquer l'existence de *"In Situ Design of Computation (IDC)"* pour les applications linéaire sur des anneaux. Nous la conception d'algorithmes pour vérifier et mettre en œuvre l'idée.

4.1 Existence d'IDC pour les applications linéaires sur des Anneaux

Il est prouvé (voire [9]) que *"In Situ Design of Computation (IDC)"* pour les applications linéaire tel que les coefficients de ces applications linéaires appartiennent à des anneaux, existent. Dans cette section nous expliquer l'idée de l'anneau $\mathbb{Z}/N\mathbb{Z}$.

Exemple 6. *Considérons une application* $E : (x_1, x_2) \longrightarrow (2x_1+3x_2, 5x_1+7x_2)$. *Les coefficients peuvent être représentés par une matrice carrée*

$$M = \begin{pmatrix} 2 & 3 \\ 5 & 7 \end{pmatrix}$$

Supposons que $N = 8 = 2^3$.
For $p^v = 2$, $(2, 2) \neq 1, \Longrightarrow m_{1,1}$ *est pas invertible.*
Parce que $\lambda_2 = 1$, *donc matrice*

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Maintenant, résolvant le système ci-dessous

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ 5 & 7 \end{pmatrix}$$

Nous avons des valeurs pour les coefficients a, b, c, et d comme ci-dessous.

$$a = -3 = 5 \ (mod\ 8)$$

$$b = -4 = 4 \ (mod\ 8)$$

$$c = 5,\ d = 7$$

Maintenant noux construisons la matrice M comme

$$M = \begin{pmatrix} 5 & 4 \\ 5 & 7 \end{pmatrix}$$

En outre,

$$G = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}$$

où,

$$G^{-1} = 5^{-1} \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix} = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix} (mod\ 8)$$

$$\begin{aligned} M' = MG^{-1} &= \begin{pmatrix} 5 & 4 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 25 & 4 \\ 25 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 4 \\ 1 & 7 \end{pmatrix} (mod\ 8) \\ U &= \begin{pmatrix} 1 & 4 \\ 0 & 1 \end{pmatrix} \\ U^{-1} &= \begin{pmatrix} 1 & -4 \\ 0 & 1 \end{pmatrix} \\ UG &= \begin{pmatrix} 1 & 4 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 5 & 4 \\ 0 & 1 \end{pmatrix} \\ M'' = M'U^{-1} &= \begin{pmatrix} 1 & 4 \\ 1 & 7 \end{pmatrix} \begin{pmatrix} 1 & -4 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 3 \end{pmatrix} \end{aligned}$$

Notez que la matrice M est décomposé en trois matrices d'affectation T, M″ et UG tel que

$$TM''UG = \begin{pmatrix} 10 & 11 \\ 5 & 7 \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ 5 & 7 \end{pmatrix} (mod\ 8)$$

Ces trois matrices d'affectation de rendement de la séquence des tâches linéaires qui calculent la cartographie donnée E et est donné ci-dessous

$$Sequence\ of\ assignments \rightarrow \begin{cases} a := 5a + 4b \\ b := a + 3b \\ a := a + b \end{cases}$$

4.2 Calcul des applications Inverse

Dans cette section, nous expliquons, par exemple, comment construire une séquence de tâches linéaire qui calcule la cartographie inverse, par une séquence de tâches linéaires sur l'anneau $\mathbb{Z}/N\mathbb{Z}$.

Exemple 7. *Soit E une application linéaire définie comme suit*

$$E\,(x_1,\,x_2,\,x_3) \longrightarrow (2x_1 + 8x_2 + 6x_3,\,3x_1 + 13x_2 + 7x_3,\,5x_1 + 5x_2 + x_3)$$

et "A" la matrice des coefficients de cartographie exprimer E, étant donné que sous

$$A = \begin{pmatrix} 2 & 8 & 6 \\ 3 & 13 & 7 \\ 5 & 5 & 1 \end{pmatrix}$$

Supposons que nous voulons effectuer des opérations dans "modulo 9". Puis

$$A = \begin{pmatrix} 2 & 8 & 6 \\ 3 & 4 & 7 \\ 5 & 5 & 1 \end{pmatrix} \ (modulo\ 9)$$

Nous construisons la séquence des tâches linéaires qui calculent l'application E, sous "modulo 9" opération, et est donnée ci-dessous

$$\left.\begin{array}{l} x_1 := 2x_1 + 8x_2 + 6x_3 \\ x_2 := 6x_1 + x_2 + 7x_3 \\ x_3 := 7x_1 + 3x_2 + x_3 \end{array}\right\}\ (modulo\ 9) \tag{S-1}$$

Nous sommes intéressés dans le calcul d'une séquence de tâches linéaire sous "modulo 9" qui calculent application inverse E^{-1}. Nous obtenons cette séquence de affectations en inversant et en réécrivant chaque affectation de la séquence S-1 (De bas en haut) qui calcule l'application E.

$$Sequence\ of\ assignments \rightarrow \left\{\begin{array}{l} x_3 := x_3 - 7x_1 - 3x_2 \\ x_2 := x_2 - 6x_1 - 7x_3 \\ x_1 := 2^{-1}\,(x_1 - 8x_2 - 6x_3) \end{array}\right. \tag{S'-1}$$

Remplacement "2^{-1}" dans la dernière affectation de la séquence S'-1 des applications par son inverse "modulo 9", nous obtenons la séquence requise des affectations.

$$\left.\begin{array}{l} x_3 := x_3 - 7x_1 - 3x_2 \\ x_2 := x_2 - 6x_1 - 7x_3 \\ x_1 := 5\,(x_1 - 8x_2 - 6x_3) \end{array}\right\} modulo\ 9$$

qui calcule la cartographie inverse. En, l'évaluation ci-dessus séquence de missions linéaires, nous obtenons l'inverse de la matrice A comme indiqué ci-dessous.

$$A^{-1} = \begin{pmatrix} 7 & 2 & 7 \\ 7 & 4 & 2 \\ 2 & 6 & 1 \end{pmatrix}$$

Vérification du produit de matrices d'affectation

La matrice

$$A = \begin{pmatrix} 2 & 8 & 6 \\ 3 & 13 & 7 \\ 5 & 5 & 1 \end{pmatrix}$$

est décomposé en trois matrices cession en vertu de modulo 9 *opération, ces matrices sont aussi donnés ci-dessous :*

$$A_1 = \begin{pmatrix} 2 & 8 & 6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} A_2 = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 13 & 7 \\ 0 & 0 & 1 \end{pmatrix} A_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 5 & 1 \end{pmatrix}$$

La produit de matrices est $A_3 * A_2 * A_1 = A$.

4.2.1 Possibilité de calculer les applications inverse

Dans cette section, nous expliquons que la construction d'une séquence de tâches linéaire qui calcule la cartographie inverse d'une cartographie donnée peut ne pas exister.

Exemple 8. *Soit E l'application linéaire définie comme ci-dessous.*

$$E\,(x_1,\, x_2,\, x_3) \longrightarrow (2x_1 + 8x_2 + 6x_3,\, 3x_1 + 13x_2 + 7x_3,\, 5x_1 + 5x_2 + x_3)$$

Supposons que nous voulons construire la séquence des tâches linéaires opération de maintien sous " modulo 12". Puis la séquence requise des missions est donné ci-dessous

$$\textit{Sequence of assignments} \rightarrow \begin{cases} x_1 := 5x_1 + 5x_2 + 9x_3 \\ x_2 := 3x_1 + 10x_2 + 4x_3 \\ x_3 := x_1 + 4x_3 \\ x_1 := x_1 + 3x_2 \end{cases}$$

Pour construire la séquence de tâches qui calcule application inverse E^{-1}*, nous avons besoin pour inverser chaque mission et de réécrire la séquence ci-dessus des devoirs de bas en haut. L'exécution de cette technique, nous obtenons la séquence des tâches comme ci-dessous.*

$$\textit{Sequence of assignments} \rightarrow \begin{cases} x_1 := x_1 - 3x_2 \\ x_3 := 4^{-1}\,(x_3 - x_1) \\ x_2 := 10^{-1}\,(x_2 - 3x_1 - 4x_3) \\ x_1 := 5^{-1}\,(x_1 - 5x_2 - 9x_3) \end{cases}$$

Observons que, le deuxième et troisième affectation de la séquence ci-dessus n'est pas inversible pour la raison que les entiers 4 *et* 10 *n'ont pas leurs inverses modulaire sous "modulo 12" opération.*

Même, l'inverse de la matrice (qui présente des coefficients de l'application E) n'a pas son inverse sous "modulo 12" car le déterminant de la matrice A

$$|A| = \begin{vmatrix} 2 & 8 & 6 \\ 3 & 13 & 7 \\ 5 & 5 & 1 \end{vmatrix} = 248$$

est pas inversible sous "modulo 12" opération.

"Par conséquent, calculer application inverse pour une application, chaque affectation linéaire dans la séquence des affectations linéaires (qui calculent le application) doit être réversible".

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 5

Conception *in situ* de calcul pour applications linéaire sur Entiers

Pour prouver l'existence de *"In Situ Design of Computation (IDC)"* pour les applications sur l'ensemble des entiers conduit à effectuer ces opérations de manière plus simple par rapport à IDC pour les applications sur des corps et des anneaux. Dans ce chapitre, nous nous concentrons à prouver l'existence de *"In Situ Design de calcul (IDC)"* pour applications sur l'ensemble de entiers.

Section 5.1 describes the existence of *"In Situ Design of Computation (IDC)"* for linear mappings over the set of integers and explains the construction of assignments involve in IDC. Section 5.2 consists in investigating bound over the number of assignments required to compute mappings by IDC. An approach with Fibonacci numbers is attempted in the same section. At the end of this section, an identity is proved that relates the determinant of matrix (presenting coefficients of mapping) to the product of coefficients of assignments involve in computing the given mappings.

5.1 Existence d'IDC pour les applications linéaires sur des entiers

Dans cette section, nous prouvons l'existence de *"In Situ Design of Computation (IDC)"* pour applications linéaires sur des entiers. L'assertion se démontre comme le théorème suivant.

Théorème 1. *Soit E une application linéaire de $\mathbb{Z}^n$. Il existe une séquence finie des affectations linéaires de la forme :*

$$
\begin{aligned}
x_{p_1} &:= x_{p_1} + f_{p_1}(x_1, \ldots, x_{p_1-1}, x_{p_1+1}, \ldots, x_n)\\
x_{p_2} &:= x_{p_2} + f_{p_2}(x_1, \ldots, x_{p_2-1}, x_{p_2+1}, \ldots, x_n)\\
\vdots &:= \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
x_{p_m} &:= x_{p_m} + f_{p_m}(x_1, \ldots, x_{p_m-1}, x_{p_m+1}, \ldots, x_n)\\
x_n &:= g_n(x_1, x_2, \ldots, x_n)\\
\vdots &:= \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
x_2 &:= g_2(x_1, x_2, \ldots, x_n)\\
x_1 &:= g_1(x_1, x_2, \ldots, x_n)
\end{aligned}
$$

qui calcule l'appliaction E, où $p_1, \ldots, p_m \in \{1, 2, \ldots, n\}$ $m, n \in \mathbb{Z}$.

"Nous estimons une borne supérieure sur le nombre d'affectations à l'aide du résultat célèbres présentés par Gabriel Lamé [39]".

Supposons que γ désigne le nombre total de missions impliquent dans le calcul application linéaire E, puis $\gamma \leq 5(n-1)d + n$, où d désigne le nombre de chiffres dans l'entrée $a(i, j)$ de la matrice A tel que $i \leq j$, but $i \neq n$, Matrice A est la matrice des coefficients de l'application E.

5.1.1 Explication et construction des affectations :

Nous expliquons la construction de missions par l'exemple 9 suivant.

Exemple 9. *Supposons que E soit l'application avec coefficients entiers comme défini par*

$$E\,(x_1,\, x_2) \longrightarrow (13x_1 + 21x_2,\, 21x_1 + 34x_2)$$

Soit la matrice M_c *représente les coefficients de l'application E.*

$$M_c := \begin{pmatrix} 13 & 21 \\ 21 & 34 \end{pmatrix}$$

Nous générons séquence des applications linéaires en effectuant les opérations des colonnes/lignes sur la matrice M_C *de telle façon qu'il en résulte des entrées entier (hors opération de division). En fait, la matrice carrée* M_C *est transformé en une matrice triangulaire, à l'achèvement de toutes les opérations. Dans le tableau 5.1, nous montrons les étapes impliquent en générant du séquence des affectations linéaire, qui impliquent dans le calcul application linéaire E.*

TABLE *5.1: Exemple 9*

Générant les affectations linéaire			
Matrice$_{(input)}$	*Opération*	*Affectation*	*Matrice*$_{(output)}$
$\begin{pmatrix} 13 & 21 \\ 21 & 34 \end{pmatrix}$	$C_2 - C_1$	$x_1 := x_1 + x_2$	$\begin{pmatrix} 13 & 8 \\ 21 & 13 \end{pmatrix}$
$\begin{pmatrix} 13 & 8 \\ 21 & 13 \end{pmatrix}$	$C_1 - C_2$	$x_2 := x_1 + x_2$	$\begin{pmatrix} 5 & 8 \\ 8 & 13 \end{pmatrix}$
$\begin{pmatrix} 5 & 8 \\ 8 & 13 \end{pmatrix}$	$C_2 - C_1$	$x_1 := x_1 + x_2$	$\begin{pmatrix} 5 & 3 \\ 8 & 5 \end{pmatrix}$
$\begin{pmatrix} 5 & 3 \\ 8 & 5 \end{pmatrix}$	$C_1 - C_2$	$x_2 := x_1 + x_2$	$\begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}$
$\begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}$	$C_2 - C_1$	$x_1 := x_1 + x_2$	$\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$
$\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}$	$C_1 - C_2$	$x_2 := x_1 + x_2$	$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$	$C_2 - C_1$	$x_1 := x_1 + x_2$	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$

Dans le tableau 5.1, première colonne présente les matrices d'entrée et la deuxième colonne présente les opérations effectuées sur colonne de ces matrices. Les affectations correspondantes sont construites dans la troisième colonne tandis matrices (modifié) après avoir effectué les opérations sont présentées dans la dernière colonne du tableau. Au lieu des opérations de la colonne, les opérations de ligne pourrait être réalisée et les affectations pourraient être construites en conséquence.

Nous mettre fin aux opérations, quand matrice carrée M_C est transformé en une matrice triangulaire, et que nous générons, la partie suivante de la séquence de affectations, directement à partir de la matrice triangulaire, écrit par la dernière rangée d'abord en tant que donnée ci-dessous.

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \longrightarrow \begin{cases} x_2 := x_1 + x_2 \\ x_1 := x_1 \end{cases}$$

Si nous ignorons la dernière attribution, cela n'affectera pas le calcul et nous aurons une séquence de huit missions linéaires au total.

Calcul de l'application inverse :

Générer l'ensemble des affectations linéaires qui calculent application inverse E^{-1}, directement à partir de l'ensemble des affectations linéaire qui calcule la cartographie E, en inversant et la réécriture de la dernière affectation à la première comme donné ci-dessous.

TABLE *5.2: Exemple 9*

Générant les affectations linéaire for E^{-1}			
Input	*Inverted*	*Evaluated*	*Coefficients*
$x_2 := x_1 + x_2$	$x_2 := -x_1 + x_2$	$x_2 := -x_1 + x_2$	$\begin{pmatrix} -1 & 1 \end{pmatrix}$
$x_1 := x_1 + x_2$	$x_1 := x_1 - x_2$	$x_1 := 2x_1 - x_2$	$\begin{pmatrix} 2 & -1 \end{pmatrix}$
$x_2 := x_1 + x_2$	$x_2 := -x_1 + x_2$	$x_2 := -3x_1 + 2x_2$	$\begin{pmatrix} -3 & 2 \end{pmatrix}$
$x_1 := x_1 + x_2$	$x_1 := x_1 - x_2$	$x_1 := 5x_1 - 3x_2$	$\begin{pmatrix} 5 & -3 \end{pmatrix}$
$x_2 := x_1 + x_2$	$x_2 := -x_1 + x_2$	$x_2 := -8x_1 + 5x_2$	$\begin{pmatrix} -8 & 5 \end{pmatrix}$
$x_1 := x_1 + x_2$	$x_1 := x_1 - x_2$	$x_1 := 13x_1 - 8x_2$	$\begin{pmatrix} 13 & -8 \end{pmatrix}$
$x_2 := x_1 + x_2$	$x_2 := -x_1 + x_2$	$x_2 := -21x_1 + 13x_2$	$\begin{pmatrix} -21 & 13 \end{pmatrix}$
$x_1 := x_1 + x_2$	$x_1 := x_1 - x_2$	$x_1 := 34x_1 - 21x_2$	$\begin{pmatrix} 34 & -21 \end{pmatrix}$

Deuxième colonne du tableau 5.2 présente série de missions linéaires qui calculent l'application inverse E^{-1} de E définie par

$$E^{-1}(x_1, x_2) \longrightarrow (34x_1 - 21x_2, -21x_1 + 13x_2)$$

Par conséquent, nous obtenons inverse de la matrice A comme indiqué ci-dessous.

$$A^{-1} := \begin{pmatrix} 34 & -21 \\ -21 & 13 \end{pmatrix}$$

$$A * A^{-1} = \begin{pmatrix} 13 & 21 \\ 21 & 34 \end{pmatrix} * \begin{pmatrix} 34 & -21 \\ -21 & 13 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

-

Possibilité de calculer les applications inverse :

IDC pour les applications inverses sur des entiers n'est pas toujours possible. Nous illustrons cela par l'exemple suivant

Exemple 10. *Considérons un application E*

$$E(x_1, x_2, x_3) \longrightarrow (2x_1 + 3x_2 + 5x_3, 3x_1 + 4x_2 - 7x_3, 8x_2 + 13x_3)$$

L'application E peut être calculée par la séquence suivante des affectations.

$$Assignments \begin{cases} x_1 := x_1 + x_2 \\ x_2 := 2x_1 + x_2 \\ x_2 := x_1 + x_2 \\ x_1 := -x_1 \\ x_1 := x_1 + x_2 \\ x_1 := x_1 + 5x_3 \\ x_3 := -x_3 \\ x_2 := x_2 + 7x_3 \\ x_3 := 24x_1 - 16x_2 + 219x_3 \\ x_2 := x_2 \\ x_1 := x_1 \end{cases}$$

La matrice des coefficients A pour la cartographie E peut être écrite comme

$$A := \begin{pmatrix} 2 & 3 & 5 \\ 3 & 4 & -7 \\ 0 & 8 & 13 \end{pmatrix}$$

and

$$A^{-1} := \frac{1}{219} \begin{pmatrix} 108 & 1 & -41 \\ -39 & 26 & 29 \\ 24 & -16 & -1 \end{pmatrix}$$

Notez que $\frac{1}{219} \notin \mathbb{Z}$ et l'affectation

$$x_3 := 24x_1 - 16x_2 + 219x_3$$

n'est pas inversible. Ainsi, l'application inverse E^{-1} n'est pas toujours calculable en inversant et la réécriture de la séquence des affectations qui calcule l'application E.

5.2 Bounds sur le nombre d'affectations :

Nous sommes intéressés à déterminer le nombre minimum de missions nécessaires au calcul de la cartographie donnée par *"In Situ Design of Computation (IDC)"* à coefficients entiers. Nous procédons par le développement des relations entre les applications qui aident à trouver nombre minimum de devoirs et d'étudier en trouvant différents exemples comptoir. Nous donnons ci-dessous une preuve qui montre que six affectations ne sont pas suffisantes pour calculer une application linéaire par IDC sur $mathbbZ^2$.

Théorème 2. *Chaque application linéaire définie comme*

$$E : (x,\, y) \longrightarrow (mx + ny,\, px + qy)$$

avec $m,\, n,\, p,\, q \in \mathbb{Z}$*, ne peut être calculée par une séquence d'au plus* 6 *affectations linéaires*

$$\left.\begin{array}{l} x := ax + by \\ y := cx + dy \\ x := ex + fy \\ y := gx + hy \\ x := ix + jy \\ y := kx + ly \end{array}\right\}$$

où $a,\, b,\, c,\, d,\, e,\, f,\, g,\, h,\, i,\, j,\, k,$ *et* $l \in \mathbb{Z}$.

5.2.1 Une approche avec les nombres de Fibonacci

Afin d'étudier le nombre minimum de affectations requises pour effectuer *"In Situ Design of Computation (IDC)"* pour les applications sur des entiers, nous établissons des relations, en gardant les nombres de Fibonacci, que les coefficients de la cartographie.

Définition 3. *Nous définissons une suite de Fibonacci-like comme*

$$F_n = F_{n-1} + F_{n-2}$$

where $F_0 = F_1 = 1$.

Définition 4. *Nous définissons une matrice de Fibonacci-like d'être une matrice de la forme*

$$\begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}$$

et les relations suivantes.

$$R_1 : \qquad \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n+1} = \begin{pmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{pmatrix}$$

$$R_2 : \qquad \begin{pmatrix} F_2 & F_3 \\ F_3 & F_4 \end{pmatrix} * \begin{pmatrix} F_{4n+1} & F_{4n+2} \\ F_{4n+2} & F_{4n+3} \end{pmatrix} = \begin{pmatrix} F_{4n+4} & F_{4n+5} \\ F_{4n+5} & F_{4n+6} \end{pmatrix}$$

Théorème 3. *Soit* $E_n : (x,\, y) \longrightarrow (F_{n-1}x + F_ny,\, F_nx + F_{n+1}y)$ *l'application sur* $\mathbb{Z}^2$*, où* F_n *est le nombre de Fibonacci. Puis l'application* E_{4k+2} *est calculé avec* $2k + 2$ *nombre des affectations, où* $k = 0,\, 1,\, 2, \cdots, n$.

Preuve. Pour $n = 2$, $E_2 = (F_1x + F_2y,\ F_2x + F_3y)$ est la matrice des coefficients est

$$A_2 = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$$

$$\text{The assignments for } A_2 \rightarrow \begin{cases} x := x + 2y \\ y := 2x - y \end{cases}$$

$$A_3 = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}$$

$$\text{The assignments for } A_3 \rightarrow \begin{cases} x := x + 2y \\ y := 3x - y \\ x := -x + y \end{cases}$$

Maintenant,

$$A_3 = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix} * A_2 = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} = A_6 = \begin{pmatrix} 8 & 13 \\ 13 & 21 \end{pmatrix} \quad \text{by } R_2.$$

Nous obtenons la séquence des affectations nécessaires au calcul de E_6 comme suit. En fait, nous emballons ensemble les missions nécessitent de calculer E_3 et les affectations nécessitent de calculer E_2.

$$\text{The assignments for } A_6 \rightarrow \begin{cases} x := x + 2y \\ y := 3x - y \\ x := -x + y \\ x := x + 2y \\ y := 2x - y \end{cases} \rightarrow \begin{cases} x := x + 2y \\ y := 3x - y \\ x := -x + 3y \\ y := 2x - y \end{cases}$$

$$\text{Similarly, since } A_3 = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix} * A_6 = \begin{pmatrix} 8 & 13 \\ 13 & 21 \end{pmatrix} = \begin{pmatrix} 55 & 89 \\ 89 & 144 \end{pmatrix} = A_{10}$$

Donc,

$$\text{The assignments for } A_{10} \rightarrow \begin{cases} x := x + 2y \\ y := 3x - y \\ x := -x + y \\ x := x + 2y \\ y := 3x - y \\ x := -x + 3y \\ y := 2x - y \end{cases} \rightarrow \begin{cases} x := x + 2y \\ y := 3x - y \\ x := -x + 3y \\ y := 3x - y \\ x := -x + 3y \\ y := 2x - y \end{cases}$$

Poursuivant dans cette voie, nous avons, $A_3 * A_{4k+2} = A_{4k+6}$

and

$$\text{The assignments for } A_{4k+6} \rightarrow \begin{cases} x := x + 2y \\ y := 3x - y \\ x := -x + y \\ x := x + 2y \\ \left.\begin{array}{l} y := 3x - y \\ x := -x + 3y \end{array}\right\} \\ \vdots \\ \left.\begin{array}{l} y := 3x - y \\ x := -x + 3y \end{array}\right\} \\ y := 2x - y \end{cases} = \begin{cases} x := x + 2y \\ \left.\begin{array}{l} y := 3x - y \\ x := -x + 3y \end{array}\right\} \\ \vdots \\ \left.\begin{array}{l} y := 3x - y \\ x := -x + 3y \end{array}\right\} \\ y := 2x - y \end{cases}$$

Ainsi, le nombre d'affectations nécessaires au calcul de la cartographie dont les coefficients sont de présenter par la matrice A_{4k+2} est $2k+2$. □

Proposition 1. *L'application linéaire*

$$E : \mathbb{Z}^m \longrightarrow \mathbb{Z}^m,\ m > 2$$

avec nombre de Fibonacci comme coefficients, défini par

$$E : (x_1, x_2, x_3, \cdots, x_n) = \begin{pmatrix} F_1x_1 & + & F_2x_2 & + & F_3x_3 & +\ldots & + & F_mx_n \\ F_2x_1 & + & F_3x_2 & + & F_4x_3 & +\ldots & + & F_{m+1}x_n \\ F_3x_1 & + & F_4x_2 & + & F_5x_3 & +\ldots & + & F_{m+2}x_n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ F_nx_1 & + & F_{n+1}x_2 & + & F_{n+2}x_3 & +\ldots & + & F_{2n-1}x_n \end{pmatrix}$$

tels que $F_n := F_{n-1} + F_{n-2}$, $n \in \mathbb{Z}$, est calculé avec $m+2$ nombre de affectation linéaire.

Ensuite, Nous prouvons une identité qui se rapporte déterminant de la matrice des coefficients au produit des coefficients d'affectations.

5.2.2 Identité :

La séquence des travaux linéaires (à coefficients entiers) qui calculent la cartographie E est une propriété intéressante pour calculer déterminant de la matrice M_c correspondant à la cartographie E. Pour explorer cette propriété, nous prouvent l'identité suivante.

Théorème 4. *Soit $E : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2$ une application linéaire définit comme $E(x, y) \longrightarrow (mx + ny,\ px + qy)$ à coefficients entiers m, n, p, $q \in \mathbb{Z}$. Soit E calculable par une séquence des affectations linéaire*

$$\left.\begin{array}{l} x := a_1x + b_1y \\ y := b'_1x + a'_1y \\ x := a_2x + b_2y \\ y := b'_2x + a'_2y \\ \vdots := \quad \vdots \quad \vdots \\ x := a_kx + b_ky \\ y := b'_kx + a'_ky \end{array}\right\}$$

où a_i, a'_i, b_i, $b'_i \in \mathbb{Z}$ sont coefficients entier pour cheque $i = 1, \ldots, k$. Puis l'identité

$$mq - pn = a_1a'_1 \ldots a_ka'_k \qquad \textit{holds.} \tag{5.1}$$

Exemple 11. *Soit $E : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2$ est application linéaire difini comme*

$$E(x, y) \longrightarrow (33x + 307y, 103x + 610y)$$

Les coefficients de l'application peut être représenté par matrice comme ci-dessous.

$$A = \begin{pmatrix} 33 & 307 \\ 103 & 610 \end{pmatrix}$$

$$\textit{déterminant } (A) = -11491 \tag{5.2}$$

L'application E peut être calculé par séquence de affectations linéaire suivent

$$Assignments \begin{cases} x := x + 9y \\ y := 3x + y \\ x := x + 3y \\ y := 2x + y \\ x := x + y \\ y := 8012x - 11491y \end{cases}$$

$$Produit\ des\ coefficients\ a_i s = -11491 \tag{5.3}$$

L'identité (théorème 4) aide à déterminer si une séquence (des affectations linéaire) qui calcule l'application inverse, existe ou non. Par exemple, pour l'application $E : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2$, une séquence qui calcule E^{-1}, s'il existe R.H.S de l'identité
refid1 devient numériquement égale à 1.

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 6

Conception *in situ* de calcul pour des applications booléenne

La recherche dans le domaine de la décomposition de la fonction booléenne est considéré comme aussi ancien que le domaine du génie circuit numérique et la décomposition des fonctions booléennes est considéré comme un problème important dans la conception de circuits logiques. Polynômes booléens produisent directement ou comme un outil dans le problème de la décomposition des fonctions booléennes. L'utilisation intensive des circuits intégrés qui comprend "modulo 2 adders", en électronique, attire l'attention vers la représentation de fonctions booléennes sous la forme de polynômes. Exprimant fonctions booléennes comme des polynômes booléens est considéré comme une méthode extensive en algèbre booléen. Méthodes polynomiales ont été employés intensivement dans la complexité du circuit et les polynômes booléens ont de nombreuses applications, par exemple, ils ont utilisé dans les codes de Reed-Muller (codes correcteurs d'erreurs). Outre un certain nombre d'applications, notamment l'exploration de texte, la découverte de connaissances, de l'ingénierie de rôle, le calcul séquentiel des application sbooléen conduit également à la décomposition des matrices booléennes et les graphes orientés.

Burckel et al. [6, 12] proposent un modèle de décomposition de applications booléen sans utiliser aucune variable supplémentaire autre que les variables disponibles en entrée. Il est prouvé que toute application linéaire booléen avec des variables d'entrée 'n' peut être calculée avec une double séquence de missions linéaires du même nombre de variables comme indiqué dans l'entrée et il est également prouvé que toute application booléen est calculé en $4n-3$ affectations [6, 12].

Dans ce chapitre, nous décrivons *"In Situ Design of Computation (IDC)"* des applications booléen. Nous commençons par expliquer les concepts de base et vérifier les résultats existants pour fournir la fondation pour complément d'enquête. Section 6.2, décrit *"In Situ Design of Computation (IDC)"* des applications booléen bijectif. Nous expliquons *"In Situ Design of Computation (IDC)"* des applications générale booléen dans la section ref Informatique générale Mappages booléenne. Dans la section 6.4, nous introduisons la méthode pour construire des *"In Situ Design of Computation (IDC)"* basée sur les polynômes construire plus de $GF(2)$ à l'égard de jeu de applications booléen. Ces polynômes conduit à construire la première affectation de la séquence, alors que, d'autres missions pourraient être construits en appliquant la méthode inductive.

6.1 Les applications Booléenne

Les applications booléenne sont très importants dans la théorie de la complexité ainsi que dans la conception de circuits et de puces pour ordinateurs numériques. Ils ont aussi un certain nombre d'applications dans différents autres domaines, notamment l'intelligence artificielle, la logique propositionnelle, génie électrique, la théorie des jeux, théorie de la fiabilité et la combinatoire. Les propriétés des applications booléen jouent un rôle crucial dans la cryptographie, en particulier dans la conception d'algorithmes symétriques clés (une classe d'algorithmes de cryptographie qui utilisent des touches de fonction booléenne à la fois pour le décryptage et le cryptage), *e.g.*, two fish, Serpent, Blowfish, CAST5, RC4, TDES, et IDEA. Polynômes booléens ont un grand nombre d'applications dans divers domaines, y compris la théorie des graphes, droit, médecine, recherche opérationnelle et de la spectroscopie [2, 19, 42, 46, 62].

Les applications booléenne décrivent comment déterminer booléenne de sortie évalués sur la base des combinaisons logiques à partir des entrées booléennes. Ces applications peuvent être représentés dans la logique propositionnelle, ou en tant que polynômes multivariés plus de $GF(2)$. Effectuer un calcul numérique en utilisant des symboles logiques est réellement mis en place par George Boole dans la théorie de la logique.

Définition 5. *Une application booléenne 'f ' est difini par*

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

L'ensemble $\{0,1\}^n$*, de tous les n-tuples* $(x_1, \cdots, x_n)$*, où chaque* x_i *est soit* 0 *ou* 1*, est la domaine pour 'f '. Il ya* 2^{2^n}*, n-ary fonctions pour chaque* n*.*

Il est important d'enquêter sur l'existence de *"In Situ Design of Computation (IDC)"* pour booléen bijectif applications pour la raison que si l'on peut calculer bijectif application booléen avec un *in situ* programme (qui se compose d'un ensemble de tâches), puis l'application inverse bijective peut également être calculée avec le même nombre de missions (qui peut être obtenue en inversant les affectations et la réécriture de bas en haut).

Ensuite, Nous expliquer et vérifier l'existence de *"In Situ Design of Computation (IDC)"* pour bijectif applications booléen.

6.2 Calcul des applications booléenne bijectives

Dans cette section, nous expliquer l'existence de *"In Situ Design of Computation (IDC)"* pour bijectif applications booléen. Nous commençons à expliquer une *in situ* programme.

Définition 6. *Un in situ programme qui calcule la application booléen de la forme*

$$E : \{0,\, 1\}^n \longrightarrow \{0,\, 1\}^n$$

consiste en une séquence de tâches d'un composant bits définis comme ci-dessous

$$x_j := f_j\,(x_1, \cdots, x_n)$$

où $f_j : \{0,1\}^n \longrightarrow \{0,1\}$ *is a linear application and* j *is the index for input variables.*

Il est prouvé [9] qui une bijection sur l'ensemble S^n est calculé par un *in situ* in situ programme impliquant $2n - 1$ nombre d'affectations à condition que n est le nombre de variables dans l'ensemble des variables d'entrée. Nous expliquons le résultat pour le cas de $S = \{0, 1\}$ comme suit.

Théorème 5. *Une application bijective E défini sur* $\{0, 1\}^n$ *peut être calculé par un in situ programme de la form.*

$$f_n,\ f_{n-1}, \cdots, f_3,\ f_2,\ f_1,\ g_2,\ g_3, \cdots, g_{n-1},\ g_n$$

Le nombre d'affectations impliqués dans le programme est $2n - 1$.

6.2.1 Propriété de linéarité

Dans cette section, nous décrivons la propriété de linéarité de l'impliquer dans les affectations *"In Situ Design of Computation (IDC)"* des applications booléenne.

Lemme 1. *Chaque application* $x_i := f_i(x_1, \cdots, x_n)$ *effectué dans un in situ programme pour calculer une application booléenne bijective doir linéaire en* x_i, *i.e.*

$$f_i(x_1, \cdots, x_n) = x_i + h(x_1, \cdots, x_{i-1},\ x_{i+1}, \cdots, x_n)$$

6.3 Calcul applications booléenne général

Dans cette section, nous décrivons les *"In Situ Design of Computation (IDC)"* dess applications booléenne général sur $\{0, 1\}^n$. Dans applications booléen général, il est possible que deux vecteurs différents peuvent avoir la même image qui n'est pas le cas pour les applications bijectives. Nous commençons par la définition suivante.

Pour plus de détails sur cette section, s'il vous plaît voir la version anglaise de la thèse. nous introduisons la méthode pour construire IDC en construisant des polynômes plus de $GF(2)$ par rapport au jeu de correspondances booléen.

6.4 IDC pour les polynômes sur GF(2)

Étude de la décomposition d'une fonction par rapport à un système donné de fonctions est un problème classique de la théorie des fonctions booléennes. Décomposition de fonction booléenne, est un problème important qui traite de répondre à des questions théoriques de la théorie des fonctions booléennes ainsi que dans des applications techniques [78]. Certaines discussions sur la forme polynomiale de fonctions booléennes et ses applications sont présentées dans [22, 66].

Notre objectif, ici, au modèle de *"In Situ Design of Computation (IDC)"* pour applications bijectif booléen qui implique polynômes sur GF(2) à l'égard de jeu de applications booléen. Il est prouvé (Lemma 1, section 6.2) que chaque affectation $x_i := f_i(x_1,\ cdots, x_n)$ impliquant, dans un *"In Situ Design of Computation (IDC)"*, pour calculer booléen applications bijectives doit être linéaire en x_i, *i.e.*,

$$f_i(x_1, \cdots, x_n) = x_i + h(x_1, \cdots, x_{i-1},\ x_{i+1}, \cdots, x_n)$$

Dans ce contexte, nous considérons le problème de tester si un polynôme construit par rapport à un ensemble d'applications booléen

$$y_1,\, y_2, \ldots, y_{i-1},\, y_i,\, y_{i+1}, \ldots, y_n$$

et un ensemble de leurs correspondants applications inverses

$$x_1,\, x_2, \ldots, x_{i-1},\, x_i,\, x_{i+1}, \ldots, x_n$$

est linéaire dans les deux y_i et x_i sur $GF(2)$. Construction des tels polynômes permettent de construire des 'IDC' pour application bijectif booléen.

Ensuite, Nous décrivons la méthode qui construisent des polynômes plus de $GF(2)$ à l'égard de ensemble de applications booléen. Nous appelons cette méthode

6.4.1 A First Tool

Dans cette section, nous introduisons la méthode pour construire les polynômes sur $GF(2)$ tel que chaque polynôme est linéaire à l'égard de la ith (par exemple) application et son correspondante image inverse. Cette méthode continue inductive-ment, sauf si elle donne 'IDC' pour l'application bijectif booléen. Supposons que nous avons une application bijectif booléen de dimension n tel que

$$\text{Set of boolean mappings} \rightarrow \begin{cases} y_1 & := f_1\,(x_1,\, x_2,\, x_3, \cdots, x_n) \\ y_2 & := f_2\,(x_1,\, x_2,\, x_3, \cdots, x_n) \\ y_3 & := f_3\,(x_1,\, x_2,\, x_3, \cdots, x_n) \\ \vdots & \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ y_i & := f_i\,(x_1,\, x_2,\, x_3, \cdots, x_n) \\ \vdots & \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ y_{n-1} & := f_{n-1}\,(x_1,\, x_2,\, x_3, \cdots, x_n) \\ y_n & := f_n\,(x_1,\, x_2,\, x_3, \cdots, x_n) \end{cases} \tag{SBM}$$

Maintenant, supposons que nous sommes à calculer un polynôme linéaire, à l'égard de la application y_i et son application inverse x_i. A cet effet, nous procédons comme suit.

Construire Applications Inverse

Comme première étape, nous calculons les images inverses pour chacune des applications booléen

$$y_1,\, y_2, \ldots, y_{i-1},\, y_i,\, y_{i+1}, \ldots, y_n$$

Correspondant à l'ensemble des applications booléennes (SBM), supposons que (SIM) l'ensemble des images inverses, comme ci-dessous

$$\text{Set of inverse images} \rightarrow \begin{cases} x_1 & := g_1\,(y_1,\, y_2,\, y_3, \cdots, y_n) \\ x_2 & := g_2\,(y_1,\, y_2,\, y_3, \cdots, y_n) \\ x_3 & := g_3\,(y_1,\, y_2,\, y_3, \cdots, y_n) \\ \vdots & \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ x_i & := g_i\,(y_1,\, y_2,\, y_3, \cdots, y_n) \\ \vdots & \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots \\ x_{n-1} & := g_{n-1}\,(y_1,\, y_2,\, y_3, \cdots, y_n) \\ x_n & := g_n\,(y_1,\, y_2,\, y_3, \cdots, y_n) \end{cases} \tag{SIM}$$

Calculer produits possibles

Dans cette étape, nous calculons tous les produits possibles des applications booléenne contenue dans l'ensemble (SBM), *i.e.*, nous calculons les produits possibles de permutations de

$$y_1,\, y_2, \ldots, y_{i-1},\, y_i,\, y_{i+1}, \ldots, y_n$$

Ces produits sont de la forme comme indiqué ci-dessous.

$$\text{Possible products} \rightarrow \begin{cases} y_1y_2,\, y_1y_3, \cdots, y_1y_n \\ y_2y_3,\, y_2y_4, \cdots, y_2y_n \\ y_3y_4,\, y_3y_5, \cdots, y_3y_n \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ y_1y_2y_3,\, y_1y_2y_4, \cdots, y_1y_2y_n \\ y_1y_3y_4,\, y_1y_3y_5, \cdots, y_1y_3y_n \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ y_1y_2y_3y_4,\, y_1y_2y_3y_5, \cdots, y_1y_2y_3y_n \\ y_1y_2y_3y_4y_5,\, y_1y_2y_3y_4y_6, \cdots, y_1y_2y_3y_4y_n \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{cases}$$

De la même, nous calculons tous les produits possibles pour l'ensemble des applications (SIM), *i.e.*, nous calculons les produits possibles de permutations de

$$x_1,\, x_2, \ldots, x_{i-1},\, x_i,\, x_{i+1}, \ldots, x_n$$

Nécessitent

Pour une application *ith* de l'ensemble (SBM) et correspondante l'application inverse de l'ensemble (SIM), l'égalité suivante doit être satisfaite

$$y_i + h_i\,(y_1,\, y_2, \ldots, y_{i-1},\, y_{i+1}, \ldots, y_n) = x_i + \acute{h_i}\,(x_1,\, x_2, \ldots, x_{i-1},\, x_{i+1}, \ldots, x_n) \tag{6.1}$$

Établissement du système

Pour satisfaire l'égalité 6.1, nous établissons un système d'équations. Pour ce faire, nous introduisons un ensemble de coefficients

$$c_1,\, c_2, \ldots, c_{i-1},\, c_{i+1}, \ldots, c_n$$

et commencer à générer un côté de l'égalité 6.1, par exemple, pour la composante *ith*, l'une des expressions possibles pourrait être de la forme

$$y_i + a_1y_2 + a_2y_3 + a_3y_2y_3 + \ldots \tag{6.2}$$

Ensuite, nous calculons les valeurs des coefficients et d'évaluer l'expression 6.2. Enfin, nous obtenons le polynôme nécessaires en substituant retour aux valeurs de ces coefficients et d'évaluer l'expression correspondante.

Nous expliquons toute la procédure dans la figure 6.1, où, 'SBM' désigne l'ensemble des applications booléen (correspondant à bijection) et 'SIM' désigne l'ensemble des images inverses correspondant à l'ensemble 'SBM'. Barre jaune montre le processus d'établissement et d'évaluer le système d'équations. Différentes étapes incluses dans le processus sont indiquées par des flèches.

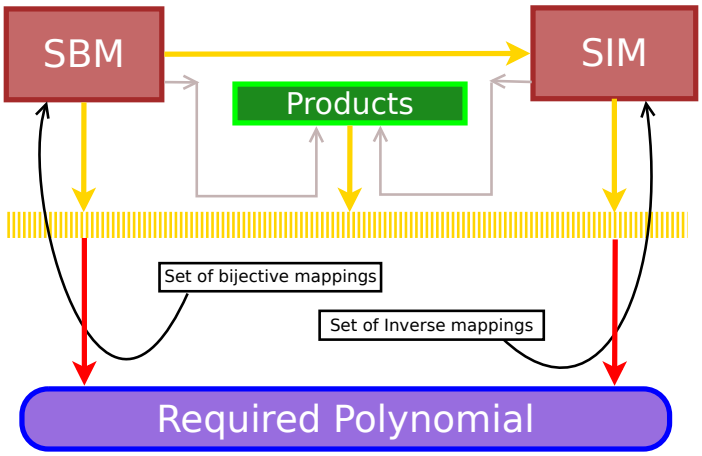


FIGURE 6.1 – "A First Tool" Interprétation

De cette façon, nous développons étape fondamentale qui mène à la construction d'IDC pour les polynômes plus de $GF(2)$.

$$y_1 + h_1\left(y_2,\, y_3, \ldots, y_{i-1},\, y_i,\, y_{i+1}, \ldots, y_n\right) := x_1 + \acute{h_1}\left(x_2,\, x_3, \ldots, x_{i-1},\, x_i,\, x_{i+1}, \ldots, x_n\right) \;\}$$

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Deuxième partie

Édition collaborative décentralisée

Chapitre 7

Système de édition collaborative

7.1 Introduction

Systems d'Edition Collaborative, fournir un environnement à un groupe d'utilisateurs, dispersés géographiquement, de sorte qu'ils peuvent éditer / modifier les mêmes données simultanément. À cette fin, des outils d'édition de collaboration ont été conçus qui permettent aux utilisateurs autorisés d'éditer / modifier un document partagé, pour voir qui d'autre travaille sur le même document et de savoir en temps réel, les changements qui sont apportés par d'autres. Les wikis, les suites bureautiques en ligne et les systèmes de contrôle de version sont les plus populaires des outils d'édition collaborative [17]. Les documents partagés sont similaires à des wikis dans le sens où un nombre d'utilisateurs peut faire modifier en ajoutant ou en supprimant le contenu. Certains des outils d'édition collaborative, facilite aux utilisateurs de communiquer par messagerie instantanée (comme une session de chat) comme un complément à l'édition collaborative. Par exemple, Wikipédia est édité par 7,5 millions d'utilisateurs et a obtenu 10 millions d'articles en seulement six ans [79].

7.1.1 Système de édition collaborative centralisée

Dans les systèmes centralisés, les messages doivent passer par un aller-retour et les utilisateurs ne peuvent pas voir leurs propres actions immédiatement. Par conséquent, les commandes locales (y compris les demandes de verrouillage) ne sont pas de réactivité. Les systèmes actuels de l'édition collaborative tels que CVS, RCS ou les wikis sont centralisées et à l'unanimité d'adopter l'architecture client-serveur. Le nœud de serveur détient une copie persistante de document partagé. Chaque poste client stocke une copie du document partagé. Un utilisateur à un poste client met à jour le document partagé par la copie locale. Toutes les mises à jour sont synchronisées avec d'autres utilisateurs à travers le nœud du serveur. Ces systèmes ne peuvent pas être adaptés aux des réseaux peer-to-peer [54].

Ci-dessous, nous discutons de certains systèmes d'édition collaborative gardant l'ordre de leur caractère restrictif sur la collaboration.

7.1.2 Système de édition collaborative décentralisée

Système de édition collaborative décentralisée fournit l'environnement, dans lequel un certain nombre d'utilisateurs situés sur plusieurs sites sont en mesure de mettre à jour objet commun (par exemple, document texte) de manière indépendante. Chaque participant au réseau a une réplique

séparée (textit ie, copie locale) du document et il / elle modifie son/sa réplique. Modification d'un texte ou un document partagé en coopération, est considéré comme un exemple bien étudié. Lorsque les utilisateurs apporter des modifications au niveau local, les répliques diverger l'une de l'autre. Les opérations effectuées sur certains sites se propager à d'autres sites et sont intégrés ou rejoué là-bas. Finalement, tous les sites d'exécuter toutes les actions des utilisateurs.

Systèmes de collaboration basée sur l'approche de transformation opérationnelle [20, 69, 79] peut être décentralisée. Les répliques pourraient ne pas converger si les utilisateurs d'exécuter des opérations dans des ordres différents. Pour la convergence, deux approches de base peut être trouvé dans la littérature. L'un d'eux est la sérialisation, *i.e.*, en appliquant un ordre total des opérations avant l'exécution [40]. Deuxième approche est la transformation opérationnelle, *i.e.*, en modifiant les paramètres des opérations à les faire courir dans des ordres différents [69]. Le deuxième approche est considérée d'être complexe et sujette à erreur, comme en témoignent les erreurs constatées dans les algorithmes publiés [53].

7.2 DCE Modèle

Notre DCE (decentralized collaborative editing) modèle est basé sur la méthode d'indexation de précision contrôlé que nous allons discuter plus tard en détail. Notre idée est applicable à tout type de documents qui portent structure séquentielle, mais, ici, pour des raisons de simplicité, nous l'illustrer en prenant un document texte commun. Pour modifier le document partagé, une insérer ou de retirer (supprimer) l'exploitation est suivie par une mise à jour pour synchroniser les répliques du document (comme dans la figure 7.1).Opération d'insertion, nécessite la collecte de renseignements sur les éléments voisins, et de l'informatique nouvel identifiant pour une entrée à insérer. Par conséquent, une action Collecter recueille de l'information et Comp va calculer nouvel identifiant. Un des avantages de cette technique est que, pour une opération d'insertion, il faut les informations que d'environ deux identifiants correspondant aux caractères voisins de la position d'insertion. Pour une opération de suppression, il exige des informations sur l'identifiant exactement correspondant à l'élément (caractère, la ligne, etc) pour être supprimés. Ceci est important car il évite les frais généraux de communication pour déplacer des éléments de données entre les machines, étant donné que les opérations de lecture sont fréquents et peuvent impliquer une grande quantité d'éléments de données. Comme toutes les répliques de documents sont mis à jour dans un ordre mondial à la fois la sérialisation de la propriété de convergence et de la propriété de préservation de causalité sont préservés. Dans ce modèle, les utilisateurs mise à jour le document partagé sans aucune restriction. Tous les conflits d'édition sont automatiquement réconciliés. Notre modèle peut être

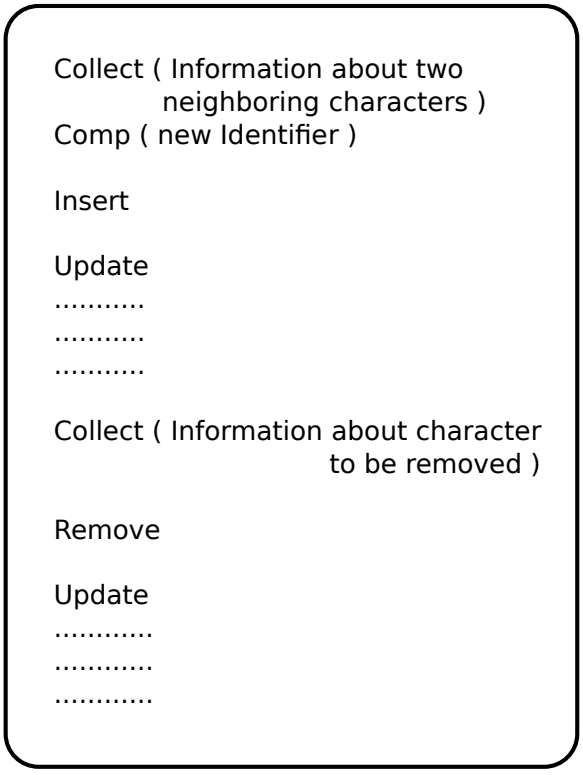


FIGURE 7.1 – Modèle

traitée comme n' importe quel jour et à tout moment-modèle. En outre, le modèle peut facilement être interprété comme un réseau $\aleph$ of n ($n \in \mathbb{N}$) utilisateurs (les sites ou les pairs) qui a attribué une unique petite valeur positive réelle $''\epsilon''$ (choisis parmi un ensemble de telles valeurs $\{\epsilon_i,\ i = 1,\ 2, \cdots n\}$, générées sous précision spécifique). Chaque utilisateur (site ou peer) de tel sorte que ces valeurs agissent comme des identifiants (site ou peer). Dans notre approche un document partagé est mappé à un intervalle $I = [a,\ b]$, où $0 \leq a < b$, for $a,\ b \in \mathbb{R}$ et des identifiants uniques sont calculées sur l'intervalle I tel que ces identificateurs uniques pourraient être assignés à des caractères ou des lignes (à éditer) dans le document partagé. Correspondant à l'insertion ou la suppression d'éléments dans le document, il existe d'insertion / délétion d'identifiants à l'intérieur de l'intervalle I. L'idée d'associer des identifiants ressemble à l'idée de mémoire tampon séquentielle partagée où chaque entrée du tampon est identifiée par l'identificateur unique [56]. Dans un document partagé, un élément peuvent présenter une séquence de caractères attribué un identifiant unique avec les propriétés suivantes :

1. Chaque élément est assignée un identifient.
2. Les deux éléments ont d'identifiants différents.
3. L'identifiant, une fois assignée ne change pas pendant toute la durée du document
4. Il existe un ordre total, $<$, sur les identifiants conformité avec l'ordre des éléments.
5. Pour deux identifiants $ID1 < ID2$, nouvel identifiant $ID3$ respecte l'inégalité $ID1 < ID3 < ID2$.

À la fin du processus d'insertion/délétion, la dernière série de unique-identifiants sera un ensemble ordonné. En fait, l'insertion d'un point de partitions de l'intervalle I en sous-intervalles. Ainsi, une insertion de $m \in \mathbb{N}$ des points dans l'intervalle I prend forme

$$a = a_0 < a_1 < a_2 < \ldots < a_{m-1} < a_m < a_{m+1} = b$$

tel que

$$a_m = a_{m-1} + \frac{a_{m+1} - a_{m-1}}{2} - \epsilon$$

en respectant les conditions de l'arrondissement et de calcul présenté à l'algorithme 2 (sera discuté en détail, plus tard dans le chapitre 8, section 8.1.3) utilisées pour créer ces identifiants.

Considérons la figure 7.2, une séquence de caractères est indexé par une séquence d'identifiants. Les caractères initiaux et finaux sont marqués avec "0" et "b", alors que les points initiaux et finaux pour les identificateurs sont marqués avec "0" and "1". Un utilisateur est destiné à insérer un caractère "m" entre deux caractères voisins "x" et "y" indexés par deux identifiants 0.292 et 0.492.. Ces identifiants sont calculées en utilisant un contrôle de précision méthode d'indexation (qui sera discuté en détail plus loin dans le chapitre 8). Nouvel identifiant 0.392 qui sert à caractère nouvel indice "m" peut être calculée par l'algorithme d'exécuter 2 (seront discutés dans le chapitre suivant).

7.2.1 Correspondance entre les éléments et les identifiants

Du point de vue d'un utilisateur, un document consiste en une séquence d'éléments (characters, lignes, etc.). Si un nouvel élément est inséré, une portion de la séquence est d'être dispersée pour créer l'espace pour le nouvel élément. De même, si un élément est supprimé, une portion

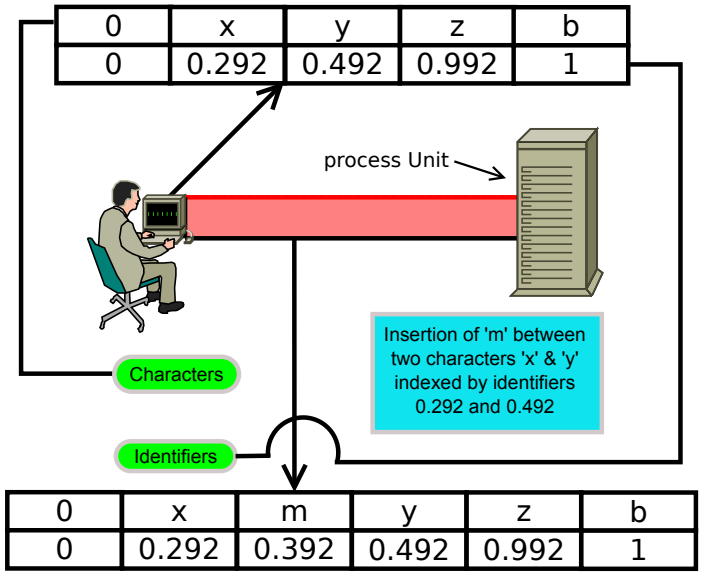


FIGURE 7.2 – Insertion de seul caractère

de la séquence est d'être fusionnés pour récupérer l'espace. Le système de montage conserve les éléments du document dans une structure de données sélectionné. L'opération de montage peuvent être mappées à la création/suppression des identifiants correspondants.

- Insérer un nouvel élément correspond à la création du nouvel identifiant.
- Supprimer un élément existant correspond à la suppression d'un identificateur.

Par conséquent, si un utilisateur modifie un document localement alors un élément correspond à la création ou la suppression d'un identificateur.

Pour maintenir les informations de commande, pour chaque élément, son avant et après les identifiants élément doivent être explicitement maintenue. Lorsque la reconstruction de contenu d'un document, le système doit suivre ces avant et après les identifiants et les cartes itérativement en leurs éléments correspondants. Un ensemble d'identifiants totalement ordonné rend facile et efficace, la lecture séquentielle et récupérer les éléments. Lorsque la reconstruction de contenu d'un document, le système doit suivre ces avant et après les identifiants et les cartes itérativement en leurs éléments correspondants. Par conséquent, le application entre les éléments et les identifiants des impacts de la performance de la lecture séquentielle des éléments et leurs identifiants correspondants.

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 8

Génération des Identifiants

Dans l'édition du système collaboratif (CES), en raison de conflits d'édition concurrente, la communication doit être construit avec soin afin de maintenir la cohérence des données. Cela conduit vers l'importance de générer des identifiants uniques qui pourraient être associés à des caractères ou de lignes à être édité. Du point de vue utilisateur d'un document est une séquence de caractères. Par conséquent, un jeu de clés éphémères qui sont sujettes à changement à la modification du document, est nécessaire pour indexer le document. Nous abordons cette question en attribuant des identifiants immuable et a ordonné aux éléments de données d'un document. Nous introduisons une Technique de précision contrôlée capable de générer un ensemble ordonné d'identificateurs uniques réelle de la même tendance. Au meilleur de notre connaissance, avant cela, il n'était pas recommandé d'utiliser un nombre réel (voir par exemple [56]) comme identifiants en raison de précision infinie. Dans ce chapitre, nous expliquons en détail la façon de générer des identifiants uniques et comment calculer la cardinalité d'un ensemble d'identifiants. La première section consiste à expliquer la technique de précision contrôlée, d'une valeur d'arrondi, l'utilisateur générant (site, par les pairs) des identifiants et des identificateurs de ligne ou de caractères, des algorithmes et des explications liées. La section 2 explique le concept de cardinalité locale ainsi que l'informatique mondiale et les cardinalités.

8.1 Création d'identifiants uniques

Dans cette section, nous expliquons, technique de précision contrôlée (PCT), que nous utilisons pour générer élément unique (ligne, caractère) identifiants ($LCID$). Puis nous illustrons algorithmes pour générer les identifiants des exemples et des chiffres.

Définition 7. *Nous définissons la précision que le nombre de chiffres après le point d'une valeur (arrondi à décimales / aux chiffres significatifs), par exemple, la précision des valeurs* 12.34600 *et* 12.345 *est* 5 *et* 3 *respectivement.*

8.1.1 Technique de précision contrôlée

Notre technique de précision contrôlée (PCT) est basé sur les hypothèses suivantes.

– $\mathbf{A}_1$: $''p_d''$ désigne la précision par défaut celle couramment utilisée par les langages de programmation sur lesquels nous effectuons des calculs.
Par exemple, en Maple, la précision utilisée est contrôlé par la variable globale "Digits"

qui a 10 que sa valeur par défaut et l'arithmétique en virgule flottante se fait en décimal avec arrondi, donc on peut ajuster la précision par défaut.

- $\mathbf{A}_2$: $''p_\epsilon''$ indique la précision prises pour les petits nombres réels positifs ϵ, qui agit comme $USIDs$.
- $\mathbf{A}_3$: $''p_r''$ indique la précision d'arrondi et est conservé au moins deux p_ϵ and p_d.

En gardant en vue, les hypothèses ci-dessus, il s'ensuit que

$$p_r < p_\epsilon < p_d \tag{8.1}$$

Nous gardons l'inégalité 8.1 comme le principe de base pour générer des identifiants uniques ($LCID$) et d'effectuer des calculs en conséquence. Dans (PCT), nous attribuons un petit nombre réel positif $''\epsilon''$ à chaque utilisateur/site ou peer (du réseau de ℵ) qui agit comme identifiant de site ou un identifiant d'utilisateur (USID). Pour créer des identifiants, les calcul sont effectuées sur la précision de $''p_d''$ présenté par barre grise dans la figure 8.1, puis les valeurs sont arrondies au cours de précision $''p_r''$ présenté par barre verte dans la figure 8.1. La barre jaune présente une précision $''p_\epsilon''$ prises pour générer l'utilisateur/site identifiants ($USIDs$).

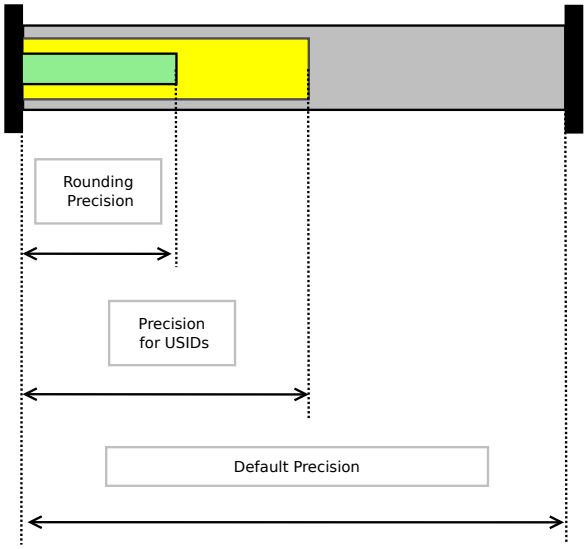


FIGURE 8.1 – Technique de précision contrôlée

Arrondi d'une valeur

Pour effectuer le calcul selon le principe de base (l'inégalité 8.1), nous introduisons la fonction qui arrondit une valeur donnée selon la définition 7. La fonction $''round(x)''$ (voir ligne 5, Fonction) arrondit une expression x au nombre entier le plus proche et la fonction $''Round\,a\,value(x,\, p_r)''$ arrondit la partie décimale d'une expression à la $p_r th$ décimale, *e.g.*, $round(-2.4)$ renvoie -2 et $''Round\,a\,value(15.0766647, 4)''$ renvoie 15.0767 exécution du calcul utilisant $Maple$ 12. Nous dénotons $''Round\,a\,value(x,\, p_r)''$ par $\left|x\right|_{p_r}$, *i.e.*, value x rounded over precision p_r by the function $''Round\,a\,value''$.

Fonction(Round a value)

Fonction : *Round a value*

Input : Value to be rounded, Desired precision
Output : Rounded Value

```
1 begin
2 |   Let dp := Desired precision;
3 |        x := Value to be rounded;
4 |   prec ⟵ 10^dp;
5 |   y = round(x * prec) / prec;
6 |   Rounded value ⟵ y;
7 |   return Rounded value;
8 end
```

8.1.2 Générant USIDs

Il ya plus d'un choix de sélectionner des valeurs qui peuvent agir en tant qu'utilisateur/site identifiants. Nous assignons chaque utilisateur/site, une epsilon (ϵ), une petite valeur positive réelle qui agit en tant que l'utilisateur / du site identifiant. Nous générons ($USIDs$) différents de la forme la forme

$$\epsilon = 0.00\ldots0_{p_r+1}d_1d_2\ldots d_{p_r}$$

$$\text{où } p_r \geq 1,\ p_\epsilon \geq p_r + \mu,\ \mu \geq 2,\ \mu \in \mathbb{N}$$

En gardant ce modèle, nous sommes en mesure de générer 9×10^w différents epsilons avec $w = p_r - 1$. Notez que pour $p_r \geq 1$, la valeur minimum de l'epsilon est $1 \times 10^{-(p_r+2)}$ est la valeur maximum est

$$\epsilon = 0.00\ldots0_{p_r+1}99\ldots9_{p_r}$$

Pour générer un ensemble de différentes $USIDs$, nous concevons l'Algorithme 1 comme ci-dessous.

Algorithme 1: Comment générer $USIDs$

Algorithme : *Generate* $USIDs$
Result : how to generate USIDs

```
1  begin
2      Let dp := Desired precision
3          𝕌 := Set of USIDs
4          w1 ⟵ dp + 2
5          w2 ⟵ 10^dp − 1
6      for i from 1 to w2 do
7              w3 ⟵ ⌊log10 (i)⌋
8              w4 ⟵ − (w1 + w3)
9          USID ⟵ i ∗ 10^w4
10         if member(USID, 𝕌) = false then
11             𝕌 := [ op(𝕌), USID ]
12             Arrange set 𝕌 in an order
13         end
14     end
15     return An Ordered Set of USIDs
16 end
```

La fonction $''member''$ (voir ligne 10, Algorithm 1) vérifie que ce soit de nouveaux calculée $USID$ existe déjà dans l'ensemble $\mathbb{U}$ ou non. Si elle n'existe pas déjà, puis "*op*" (voir ligne 11, Algorithm 1) le recueille comme un membre de l'ensemble $\mathbb{U}$. L'algorithm 1 renvoire renvoie un ensemble de différents epsilons sous la précision p_r. Par exemple, $Generate\ USIDs(4)$ renverra un ensemble de 9000 différents epsilons calculés sous la précision $p_r = 4$ avec la valeur minimum de l'epsilon, égales 1×10^{-6} et avec valeur maximum égales 0.000009999. Command *op* est utilisé à ecrir est utilisé pour écrire l'entrée dans une rangée. Les $USIDs$ maintiennent la propriété $|USID|_{p_r} = 0$.

8.1.3 Le procédure

Il est connu que la formule classique de milieu calcule milieu de deux points donnés a et b comme $(a+b)/2$ et si nous l'appliquons à calculer tous les milieux possibles d'un intervalle, alors il travaille de façon itérative et continue à la partition de l'intervalle infini. Pour calculer à chaque fois différents et finie milieux possibles pour un intervalle, elle nécessite quelques modifications. Soit n ($n \in \mathbb{N}$) utilisateurs calculer tous milieux possibles sur un intervalle I, nous intéresse, que :

- ⋆ Points calculés par un utilisateur doit être différent de points calculés par tous les autres
- ⋆ Ensemble des points calculés par chaque utilisateur doit être un ensemble ordonné
- ⋆ Ensemble des points calculés par chaque utilisateur doit être fini.

Pour répondre à ces exigences, nous faisons quelques hypothèses et de modifier la formule du point milieu classique, pour un intervalle $I = [\,a,\, b\,]$ avec $0 \leq a < b$ tel que $\forall\ x,\, y \in I$ avec $x < y$,

comme ci-dessous

$$f(x, y) = x + \frac{y - x}{2} - \epsilon \tag{8.2}$$

où ϵ est un des $USIDs$. Notez que le point calculé en ce façon ne sera pas équidistant des points x et y (comme expliqué en Figure 8.2) en raison des conditions posées pour le calcul. Dans toutes

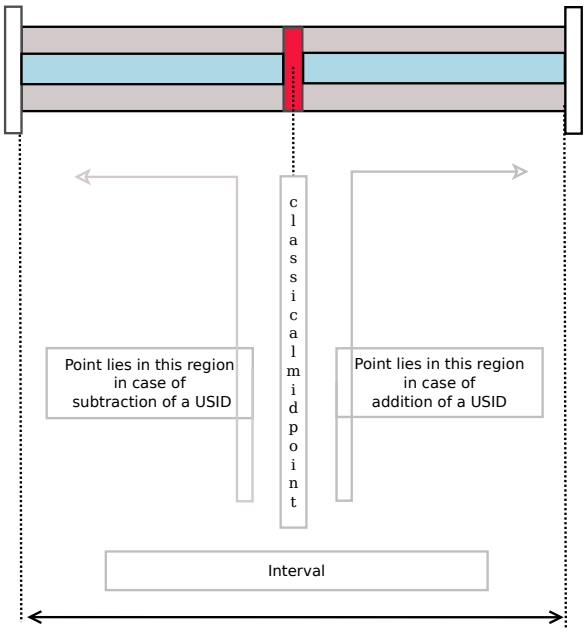


FIGURE 8.2 – Position d'un identifiant dans l'intervalle

nos expérimentations, nous prenons l'intervalle que $I = [\,0,\, 1\,]$ mais nous ne sommes pas limités à ne prendre que cet intervalle. Pour les deux points $a_1,\ b_1$, nous mettons en œuvre la formule (8.2) en concevant Fonction *middle*, avec $USID_{max} = max(USIDs)$. La Fonction *middle* prend deux points en entrée et retourne un nouveau point en appliquant la formule (8.2) suivant la technique de précision contrôlée (PCT).

Nous introduisons des conditions supplémentaires pour assurer la préservation de commande et d'obtenir les points calculés sous la forme d'un ensemble ordonné. Pour ce faire, nous concevons Algorithme 2 qui contient Fonction *middle* comme une partie de celui-ci.

Fonction *middle*(Comment génerer nouvel identifiant)

Fonction : *middle*

Input : I=[a_1, b_1], $USID$, $USID_{max}$, Desired precision
Output : A real value

begin
  Let dp := Desired precision;
  $a \longleftarrow \left|a_1\right|_{dp}$;
  $b \longleftarrow \left|b_1\right|_{dp}$;
  $\nabla := \left|\frac{b-a}{2}\right|_{dp}$;
  if $\nabla > USID_{max}$ **then**
    $point := \left|a + \nabla\right|_{dp} - (USID)$;
  end
  return *point*;
end

Algorithme 2: Comment générer ensemble des $LCIDs$

Algorithme : *Generate LCIDs*

Input : $Interval$, $USID$, $USID_{max}$, Desired precision
Output : Set of unique identifiers (LCIDs)

begin
  Let $res := Interval$
    $dp :=$ Desired precision
  $tempres :=$ Set of $LCIDs$
  while $nops(tempres) \neq nops(res)$ **do**
    if $nops(tempres) \neq 0$ **then**
      $res \longleftarrow tempres$
      $tempres \longleftarrow \phi$
    end
    for $i \longleftarrow 1$ *to* $nops(res) - 1$ **do**
      $LCID \longleftarrow middle(res[i], res[i+1], USID, USID_{max}, dp)$
      $tempres := [\, op(tempres), res[i] \,]$
      if $(LCID \neq res[i]$ *and* $LCID \neq res[i+1])$ **then**
        verify that
        if $(LCID < res[i+1]$ *and* $LCID > res[i])$ **then**
          $tempres \longleftarrow [\, op(tempres), LCID \,]$
        end
      end
    end
    $tempres \longleftarrow [\, op(tempres), res[nops(res)] \,]$
  end
  return *Set of LCIDs*
end

Dans l'Algorithme 2, le fonction *″nops″* (voir ligne 5, 6 et 10) utilisé à calculer les nombre des éléments dans l'ensemble. Comme une exemple, $nops([5, 6, 7])$ retourn 3, qui est le nombre d'éléments.

Ensuite, nous résumons toutes les étapes qui sont effectuées ci-dessus, dans le calcul $LCIDs$.

Outlines

Sit l'interval $I = [a, b]$, $0 \leq a < b$, present un document videt ou a et b sont deux $LCIDs$ pour deux characters/lines correspondent. Pour calculer une nouvelle $LCID$ (say p) entre a et b, nous procédons comme suit :

C1. Round a and b over the precision p_r

C2. Evaluate $\nabla = \frac{|b|_{p_r} - |a|_{p_r}}{2}$ and round it over p_r

C3. If $\nabla > USID_{max}$ then go to the next step

C4. $p := \big||a|_{p_r} + \nabla\big|_{p_r} - USID$

C5. Verify that $p \not\leq a$ and $p \not\geq b$.

8.2 Computing cardinalité

La cardinalité d'un ensemble est une mesure du "nombre d'éléments de l'ensemble". Par exemple, l'ensemble $A = \{2, 4, 6\}$ contient 3 éléments, et donc A a une cardinalité de 3. Technique de précision contrôlée (PCT) permet de calculer le nombre de $LCIDs$, qui peuvent être générés en prenant une précision particulière.

Définition 8. *Le nombre possible d'identifiants (LCIDs) qui pourraient être créés pour un utilisateur (le site) sous une précision particulière est appelé cardinalité locale et est notée C_l.*

8.2.1 Local cardinalité

Soit γ denote la longueur de l'intervalle $I = [a, b]$, $0 \leq a < b$ puis $\gamma = |b - a|$ avec $\gamma > 0$. pour le précision p_r, cardinalité locale pour un seul utilisateur peut être calculé comme

$$C_l = \begin{cases} \gamma \times 10^{p_r} & \text{if } b > a \geq 0, \text{ for } a, b \in \mathbb{N} \\ (\gamma_1 \times 10^{p_r}) - 1 & \text{if } b > a > 0, \text{ for } a, b \in \mathbb{R} \\ & \text{and } \gamma_1 = |\gamma|_{p_r} \end{cases}$$

tel que

$$p_r < p_\epsilon < p_d$$

$USIDs$ sont du modèle, comme expliqué dans la section 8.1.2.

Exemple 12. *Supposons que $I_1 = [0, 1]$, $p_r = 3$, $USID_{min} = 0.00001$, et $USID_{max} = 0.0000999$. La cardinalité locale d'un ensemble de $LCIDs$, qu'un seul utilisateur avec $USID = 0.0000439$ peut générer, est calculé par $C_l = 1 * 10^3 = 1000$. Maintenant, soit le même utilisateur veut calculer la cardinalité locale de $LCIDs$ entre deux $LCIDs$ (déjà calculé par lui). Soit $I_2 = [0.7609561, 0.8139561]$ puis $\gamma_1 = 0.053$ et $C_l = (0.053 * 10^3) - 1 = 52$.*

Définition 9. *Dans un réseau d'utilisateurs / pairs, l'union de cardinalité locale calculée par / à chaque utilisateur / par les pairs est appelé cardinalité globale et est notée* C_g.

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 9

Propriétés et analyse

Certaines propriétés importantes de l'ensemble des identifiants $LCID$ généré par l'algorithme 2 (voir le chapitre 8, section 8.1.3) conduit vers les avantages de la technique de précision contrôlée. De même, une analyse de la fonction 8.2 aide dans l'exploration de l'utilité de la fonction dans des conditions d'arrondi et ses limites. Dans ce chapitre, nous discutons quelques propriétés importantes qui tiennent sur l'ensemble des $LCIDs$ calculé par l'algorithme 2. Nous faisons la comparaison des fonctions 8.2 avec la formule classique de milieu dans des situations différentes. Nous analysons algorithme 2 en modifiant et en appliquant des conditions différentes d'arrondi sur la fonction 8.2.

9.1 Propriétés

Dans cette section, nous discutons de la propriété bijection bien connu et propriétés de fermeture qui sont valables sur l'ensemble des identificateurs.

9.1.1 Propriété bijection

Proposition 2. *Soit S' est l'ensemble defini sur l'interval $I = [\,a,\, b\,], 0 \leq a < b$, comme*

$$S' = \{\frac{n'}{10^{p_r}} \mid n' \in \mathbb{N},\, a < n' \leq \beta,\, \beta = \gamma * 10^{p_r}\}$$

où γ est l' langeur de intervallele I avec $|\gamma|_{p_r} > 0$. Pour $1 \leq k \leq n,\ n \in \mathbb{N}$, soit ϵ_k est un des $USIDs$ puis l'fonction $f_k : S' \longrightarrow S''$ defini par $f_k(x') = x' - \epsilon_k,\, \forall\, x' \in S'$, est un bijection.

Preuve. La domainne de la fonction f_k est l'ensemble

$$S' = \{\frac{n'}{10^{p_r}} \mid n' \in \mathbb{N},\, a < n' \leq \beta,\, \beta = \gamma * 10^{p_r}\}$$

et le range de la fonction f_k est l'ensemble

$$S'' = \{x' - \epsilon_k \mid \forall\, x' \in S',\, p_\epsilon \geq p_r + \mu,\ \mu \geq 2\}$$

Maintenant, nous devons montrer que la fonction donnée f_k est injective et surjective

– Injection
Supposons que

$$x'_1 \neq x'_2,\ \forall\ x'_1,\ x'_2 \in S'$$

puis

$$f_k(x'_1) = x'_1 - \epsilon_k \text{ and } f_k(x'_2) = x'_2 - \epsilon_k$$

Let

$$f_k(x'_1) = f_k(x'_2)$$
$$\implies x'_1 = x'_2$$

Une contradiction à supposer que $x'_1 \neq x'_2$. Ainsi

$$f_k(x'_1) \neq f_k(x'_2)$$

– Surjection
Pour surjection, Nous devons à prouver que

$$\exists\ x' \in S',\ \text{such that } x'' = f_k(x'),\ \forall x'' \in S''$$

$$\because\ \ x' = x'' + \epsilon_k$$
$$\therefore\ \ f_k(x'' + \epsilon_k) = x'' + \epsilon_k - \epsilon_k = x''$$
$$\implies f_k \text{ is a surjective functions}$$

Dorénavant, pour $1 \leq k \leq n$, f_k present une famil de bijective fonctions. □

Définition 10. *Nous denotons, l'ensemble des LCIDs calculé par utilisateur U_k avec $USID\ (\epsilon_k)$ sur l'intervalle $[a,\ b]$, $0 \leq a < b$, by S_k, où $1 \leq k \leq n,\ n \in \mathbb{N}$.*

Corollaire 1. *Pour l'intervalle $I = [\,a,\ b\,]$, $0 \leq a < b$, deux ensembles S'' et S_k sont egale.*

Preuve. Pour un $USID = \epsilon_k$, l'ensemble S_k peut être ecrir comme

$$S_k = \{x_k \mid x_k = |x_k|_{p_r} - \epsilon_k\}$$

Soit $\acute{x_k} \in S_k$ puis $\acute{x_k} = |\acute{x_k}|_{p_r} - \epsilon_k$. Maintenant $|\acute{x_k}|_{p_r} \in \acute{S}$, où

$$S' = \{\frac{n'}{10^{p_r}} \mid\ n' \in \mathbb{N},\ 1 \leq n' \leq \beta,\ \beta = \gamma \times 10^{p_r}\}$$

Mais $|\acute{x_k}|_{p_r} - \epsilon_k \in S''$ by Proposition 2.
Par conséquen S_k et S'' sont des ensembles égale généré sur la même intervalle. □

9.1.2 Propriétés de clôture

Lemme 2. *Soit pour$1 \leq k \leq n,\ n \in \mathbb{N},\ \epsilon_k \in \{\epsilon_1,\ \epsilon_2,\ \epsilon_3, \ldots, \epsilon_{n-1},\ \epsilon_n\}$ est $USID$ pour utilisateur U_k et S_k soit l'ensemble des LCIDs généré pour utilisateur U_k sur l'intervalle $I = [\,a,\ b\,]$, $0 \leq a < b$. Puis pour $x,\ y \in S_k$, $x < y$,*

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in S_k$$

Preuve. Soit $x,\ y \in S_k$ puis

$$x = |x|_{p_r} - \epsilon_k,\ \text{et}\ |x|_{p_r} \in \acute{S}$$
$$y = |y|_{p_r} - \epsilon_k,\ \text{et}\ |y|_{p_r} \in \acute{S}$$

où

$$S' = \{\frac{n'}{10^{p_r}} \mid\ n' \in \mathbb{N},\ 1 \leq n' \leq \beta,\ \beta = \gamma \times 10^{p_r}\}$$

Supposons que

$$\nabla = \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r} > (USID)_{\max}$$

C'est-à-dire, la condition C5. (section 23) est satisfaite. Il donne $\nabla \in \acute{S}$, Maintenant

$$|x|_{p_r} < \left||x|_{p_r} + \nabla\right| \leq |y|_{p_r}$$

Mais

$$\left||x|_{p_r} + \nabla\right|_{p_r} \in \acute{S}$$

Par conséquent, par définition 10.

$$\left||x|_{p_r} + \nabla\right|_{p_r} - \epsilon_k \in S_k$$

□

Lemme 3. *Soit S_i, S_j, et S_k des ensemble des LCIDs généré par utilisateurs U_i, U_j et U_k avec USIDs, ϵ_i, ϵ_j et ϵ_k sur l'intervalle $I = [a,\ b]$, $0 \leq a < b$ respectivement. Puis, pour chaque deux points $x \in S_i$, $y \in S_j$, $x < y$ et pour $\epsilon_j < \epsilon_k$.*

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in S_k$$

Preuve. Soit $x \in S_i$ et $y \in S_j$ puis

$$x = |x|_{p_r} - \epsilon_i,\ \text{et}\ |x|_{p_r} \in \acute{S}$$

$$y = |y|_{p_r} - \epsilon_j,\ \text{et}\ |y|_{p_r} \in \acute{S}$$

Aussi, les ensembles $S_i,\ S_j, \text{et}\, S_k$ peut ecrire comme

$$S_i = \{x' - \epsilon_i \mid\ \forall\ x' \in S',\ p_\epsilon \geq p_r + \mu,\ \mu \geq 2\}$$

$$S_j = \{x' - \epsilon_j \mid\ \forall\ x' \in S',\ p_\epsilon \geq p_r + \mu,\ \mu \geq 2\}$$

$$S_k = \{x' - \epsilon_k \mid\ \forall\ x' \in S',\ p_\epsilon \geq p_r + \mu,\ \mu \geq 2\}$$

où

$$S' = \{\frac{n'}{10^{p_r}} \mid\ n' \in \mathbb{N},\ 1 \leq n' \leq \beta,\ \beta = \gamma \times 10^{p_r}\}$$

Supposons que

$$\nabla = \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r} > \epsilon,\ \text{avec},\ \epsilon = max(\epsilon_i,\ \epsilon_j,\ \epsilon_k)$$
$$\Longrightarrow |x|_{p_r} \neq |y|_{p_r}$$

Sinon, condition C5. (section 23) ne pas être valide.
Maintenant, soit

$$\acute{p} = \big||x|_{p_r} + \nabla\big|_{p_r}$$

puis

$$|x|_{p_r} < \acute{p} \leq |y|_{p_r}$$

Soit

$$\acute{p} = |y|_{p_r}$$

puis

$$\acute{p} \in \acute{S}$$

Maintenant,

$$\epsilon_j < \epsilon_k$$

Par conséquent,

$$\acute{p} - \epsilon_k < \acute{p} - \epsilon_j \text{ and } \acute{p} - \epsilon_k \in S_k$$

□

Lemme 4. *Pour $1 \leq k \leq n$, $n \in \mathbb{N}$, soit U_k l'ensemble des utilisateurs/sites et $\mathbb{S} = \bigcup_{k=1}^{n} S_k$, avec $x < y$ l'ensemble des LCIDs génère par tous les U_k/sites puis pour $x, y \in \mathbb{S}$ et $\epsilon_k \in \{\epsilon_1, \epsilon_2, \epsilon_3, \ldots, \epsilon_{n-1}, \epsilon_n\}$, le point*

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in S_k$$

Preuve. Supposons que

$$\nabla = \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r} > \epsilon,$$

avec

$$USID_{max} = max(\epsilon_1, \epsilon_2, \epsilon_3, \ldots, \epsilon_{n-1}, \epsilon_n)$$

Maintenant, soit $x,\ y \in S_k$ puis, par Lemme (2).

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in S_k$$

Soit $x \in S_i$ et $y \in S_j$ puis, par Lemme (3).

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in S_k$$

Finallement, soit $x,\ y \in S_i$

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in S_k$$

Parce que $S_k \subseteq \mathbb{S}$, donc

$$\left||x|_{p_r} + \left|\frac{|y|_{p_r} - |x|_{p_r}}{2}\right|_{p_r}\right|_{p_r} - \epsilon_k \in \mathbb{S}$$

D'où la preuve. □

9.2 Analyse

Dans cette section, nous faisons une analyse de la fonction 8.2 d'explorer certains de ses avantages. La formule du point milieu classique, exclut la condition C4. (voir pour les détails, le chapitre 8, section 8.1.3).

C4. $p := \left||a|_{p_r} + \nabla\right|_{p_r} - USID$

et les restrictions de arrondissement appliquées dans Outlines (chapitre 8, section 8.1.3). Nous faisons une comparaison entre deux formules :

- Comparaison sous des conditions d'arrondi
- Comparaison sans des conditions d'arrondi

9.2.1 Comparaison sous arithmétique en virgule flottante

Nous préférons également d'utiliser la fraction

$$x_i + \frac{x_{i+1} - x_i}{2} \text{ sur } \frac{x_{i+1} + x_i}{2}$$

car le milieu qu'elle calcule peut être absent de l'intervalle à cause de certaines restrictions dûes à l'arithmétique flottante. Par exemple, pour un intervalle $I = [0.982, 0.987]$, si nous sommes limités à effectuer des calculs sur une machine avec trois décimales arithmétiques chiffres points hachées flottant, alors

$$x + y = 1.96 \text{ et } \frac{x + y}{2} = 0.980 \notin I$$

9.2.2 Effet des conditions différentes d'arrondi

Dans cette section, nous mettons en œuvre la fonction 8.2 dans des conditions différentes d'arrondi afin que l'effet plus de $LCID$ peut être observée. Nous considérons la situation dans laquelle les utilisateurs/les sites du réseau de participer à une afin de calculer $LCID$. Chaque utilisateur du réseau calcule $LCID$ localement et l'envoie à tous les autres utilisateurs/sites. Pour un ensemble de $USIDs$, $\{\epsilon_1, \epsilon_2, \epsilon_3, \ldots, \epsilon_{n-1}, \epsilon_n\}$, soit $USID_{\min} = min(\epsilon_k)$, où $1 \leq k \leq n$, for $n \in \mathbb{N}$, Pour observer l'effet des conditions de l'arrondissement sur la fonction 8.2, nous modifions la fonction *middle* et observer l'effet correspondant.

9.2.3 Assurance de Préservation d'Ordre

Généralement, un point central, de deux points, calculée par la formule classique de milieu se trouve dans ces deux points, mais il ne peut pas vrai sous l'environnement de certains, comme expliqué dans la section 9.2.1, où un point médian n'a pas contenue dans le intervalle. Dans générer les identifiants, problème d'ordre préservant existent spécifiquement lors de l'échange des mises à jour et dans les identificateurs informatique basée sur des points distants (identifiants), nous avons discuté de ce problème en détail dans la section 1.2.2 et section 1.2.3.. Pour faire face à ce problème, nous posons la condition C5. (voir, contours, section 8.1.3) afin que la propriété de préservation commande peut être assurée.

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Chapitre 10

L'échange de points et la variation dans cardinalité global

Pour un réseau ℵ comporte n utilisateurs/sites/peers, échange de information à travers le réseau ℵ exige l'échange de $LCIDs$ entre utilisateures/sites/peers dans le réseau pendant l'édition collaboratif
Nous nous intéressons à concevoir un modèle à ce réseau d'une telle façon que cardinalités global s'approche à son supérieur attaché. Nous nous concentrons dans deux différentes situations comme suit

1. Échange retardé de points
2. Échange rapide de points

Dans ce chapitre, section 10.1 traite de la première situation tandis que section 10.2 discute de la deuxième situation en détail. Pour l'ensemble de la communication dans le réseau, les identifiants sont générés en utilisant les critères (comme expliqué dans la section 8.1.3, Outlines, page 59) respectant les conditions de précision posées.

10.1 Échange retardé de points

Supposons qu'un utilisateur U_1 commence points de calcul et l'envoyer à tous les autres utilisateurs (disons paresseux), alors toute autre commence l'informatique, après avoir reçu des points de U_1, en raison du fait que d'autres sont paresseux. Maintenant tout autre utilisateur qui veut calculer les points doivent avoir à vérifier la validité des conditions requises (section 8.1.3, Outlines, chapitre 8) sur les points y compris les points qu'il / elle a reçu avant de commencer le calcul. Cela peut affecter la cardinalité globale aux approches de sa borne supérieure. nous expliquer cet environnement en considérant deux situations différentes, d'abord lorsque les utilisateurs participent de manière aléatoire et d'autre part quand ils participent à une commande. Nous décrivons le premier cas par un modèle stochastique de type et la deuxième par un modèle déterministe-like.

10.1.1 Participation au hasard

Modèle 1

Nous introduisons ce modèle pour le réseau de *aleph* sous les conditions suivantes.

1. Nous produisons d'un ensemble ordonné (augmentant) $\mathbb{U}$ des utilisateur/site-identifiants ($USIDs$) selon le précision sélectionné.
2. De l'ensemble $\mathbb{U}$, on choisit un utilisateur/site-identifiant au hasard, *i.e.*, n'importe quel utilisateur dans le réseau de *aleph* de participer au hasard.
3. L'utilisateur calcule tous les points possibles (selon les critères (outlines section 8.1.3, page 59)) et envoyer ces points, à la fois, à d'autres, *i.e.*, un paquet de points d'envoyer et de recevoir au cours du processus.
4. Tous les utilisateurs du réseau sont en mesure de recevoir les points qui ont envoyé par d'autres.
5. Chaque utilisateur de vérifier la validité des conditions requises (section 8.1.3, Outlines, chapter 8) sur les points y compris les points reçus par l'utilisateur.
6. Si un utilisateur calcule de nouveaux points, puis cardinalité globale sera mise à jour sinon elle reste inchangée.
7. Tout utilisateur peut participer à plus d'une fois, cependant, l'insertion de points dépendra de la validité des conditions requises.

10.1.2 participation à l'ordre

Modèle 2

Nous intr Nous introduisons ce modèle pour le réseau $\aleph$ sous les conditions suivantes.

1. Nous générons un ensemble ordonné $\mathbb{U}$ (en augmentation) d'utilisateur/site/peer identifiants ($USIDs$) selon la précision choisie.
2. De l'ensemble $\mathbb{U}$, nous choisissons l'utilisateur identifiant dans l'ordre croissant, *i.e.*, l'utilisateur suivant a $USID$ supérieur au $USID$ de l'utilisateur précédent.
3. L'utilisateur calcule tous les points possibles suite aux critères (outlines section 8.1.3, page 59) et envoie ces points, immédiatement, à d'autres, *c-à-d*, un paquet de points envoient et reçoivent pendant le processus.
4. Tous les utilisateurs dans le réseau sont capables de recevoir les points qui envoyé par d'autres.
5. Chaque utilisateur vérifie la validité des conditions exigées (section 8.1.3, Outlines, chapitre 8) sur les points en incluant les points reçu par l'utilisateur.
6. Si un utilisateur calcule de nouveaux points alors cardinalité global sera actualisé autrement cela restera inchangé.
7. Un utilisateur peut participer seulement une fois.

10.2 Échange rapide de points

Nous proposons un autre modèle pour le réseau $\aleph$ d'utilisateurs qui est en activité d'une meilleure façon de maximiser cardinalité global.

Modèle 3

Ce modèle est conçu avec les hypothèses suivantes.

1. Nous générons un ensemble ordonné d'utilisateur / site / peer-identifiants (USIDs) $\mathbb{U}$ par rapport à la précision choisie.
2. De l'ensemble $\mathbb{U}$, nous choisissons l'utilisateur identifiant dans l'ordre croissant, *i.e.* utilisateur suivant a user-identifier plus grande que l'utilisateur d'identifiant utilisateur précédent
3. L'utilisateur calcule tous les points possibles suite aux critères (outlines section 8.1.3, page 59) et envoyer ces points immédiatement sans n'importe quel retard, à d'autres le présent dans le réseau, *c'est-à-dire'*, un utilisateur dans le réseau calcule tous les points possibles et envoyer ces points à d'autres avant de recevoir des points d'autres
4. Tous les utilisateurs dans le réseau sont capables de recevoir les points qui envoyé par d'autres.
5. Chaque utilisateur vérifie la validité des conditions exigées sur les points produits par lui/son.
6. Si un utilisateur calcule de nouveaux points cardinalité alors global sera actualisé autrement cela restera inchangé.
7. Un utilisateur peut participer seulement une fois.

10.2.1 comparaison

Nous comparons toutes les situations que nous avons déjà discutées. Dans Modèle 1, tous les utilisateurs ne sont pas capables d'actualiser le dû cardinalité global à la raison cette Condition C5. n'est pas valide pour tous les utilisateurs. Pour exemple, l'utilisateur avec le $USID = 0,00099$ succédé à insérer le 100 points et actualise cardinalité global du 402 au 502 mais d'autre part l'utilisateur avec le $USID = 0,00044$ n'a pas réussi pour insérer n'importe quel point et finalement, il n'y a aucune mise à jour dans cardinalité. global
Modèle 2, présente la situation paresseuse, *c-à-d*, un utilisateur avec le $USID = 0,0001$ commence à calculer des points et envoyer aux utilisateurs suivants en réseau $\aleph$. Parce que les utilisateurs sont assignés le $USID$ dans un ordre augmentant et ils suivent cet ordre dans le fait d'envoyer les points. Donc les Critères 8.1.3, page 59) en incluant la Condition C5. tous les deux restent valides pour chaque utilisateur en réseau $\aleph$. Il permet à chaque utilisateur d'accomplir son cardinalité local, finalement, chaque utilisateur actualise cardinalité. global
Le plus pire cas est où seulement un utilisateur est capable d'insérer des points, mais tous les autres manquer d'actualiser cardinalité global. Dans ce cas-là, les critères sont satisfaits pour tous les utilisateurs mais la Condition C5. n'est pas satisfait.
Une situation idéale dans le fait de maximiser cardinalité global accomplit en exécutant Modèle 3. La cardinalité globale obtenue dans Modèle 2 et dans Modèle 3 est similaire, mais dans Modèle 2, les utilisateurs doivent suivre l'ordre imposé dans l'informatique et le fait d'envoyer les points, mais Modèle 3 (section 10.2) est indépendant de telles restrictions.

Pour plus de détaille, Veuillez consulter la version anglaise de la thèse, SVP!

Troisième partie

Évaluation

Chapitre 11

Travaux à venir et conclusion

Dans cette thèse, nous avançons les capacités de calcul séquentiel et de systèmes distribués en enquêtant *"In Situ Design of Computation (IDC)"* (la partie 1) et en développant des méthodes pour faciliter la communication dans les systèmes révisants en collaboration décentralisés. Les contributions clé pourraient être résumées comme

- Décomposition Séquentiel des opérations par *"In Situ Design of Computation (IDC)"*.
- La décomposition polynomiale sur GF(2).
- Conception système décentralisé de l'édition collaborative.

Ce dernier chapitre présente les travaux futurs et d'autres perspectives de résultats.

11.1 Travaux à venir

Nous commencerons par discuter des orientations possibles des recherches futures et identifier les questions ouvertes qui ont émergé de cette thèse.

11.1.1 La dispersion

Dans l'édition collaborative, le processus d'échange de points correspond à l'insertion / suppression de points lors de l'édition / modification de documents partagés. La suppression du point crée la possibilité d'insérer un nouveau point. Mais à chaque retrait de points ne garantit pas l'insertion du nouveau point comme on peut le désir. Nous décrivons ce problème comme suit.

Description du problème

Considérons la Figure 11.1. Supposons que (après l'exécution d'une expérience), nous avons un document partagé avec, par exemple, neuf points et plus de points peuvent être ajoutés.

$$[\, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9 \,]$$

Un des utilisateurs a tenté d'insérer un point à la position entre les points p_4 et p_5 et exécute Algorithme 2 pour effectuer la tâche, mais la demande n'est pas acceptée. Ensuite, supposons que une / deux utilisateur / site / utilisateurs / sites supprime / supprimer deux points p_3 et P_7 et mises à jour sont exécutées. Un autre utilisateur / même / site à nouveau tenté l'insertion

d'un point à la position entre les points de p_4 et P_5. Mais toujours la demande n'est pas acceptée par l'algorithme 2 raison de l'échec de conditions.

Supposons que nous sont donnés avec un mécanisme de dispersion qui peut déplacer les points de telle sorte que l'espace disponible / mémoire peut être pleinement utilisé. Maintenant, appliquer le mécanisme de dispersion de créer l'espace pour faciliter l'intention de l'utilisateur/site. Après l'exécution de mises à jour, les points mp_4, mp_5 et mp_6 sont les points modifiés.
Notons que, un utilisateur / site tenté l'insertion de points à la même position et la demande est maintenant accepté par l'algorithme 2.
En fait, nous sommes intéressés à concevoir un mécanisme de dispersion complète, qui permet de récupérer des espaces vides (ou slots mémoire).

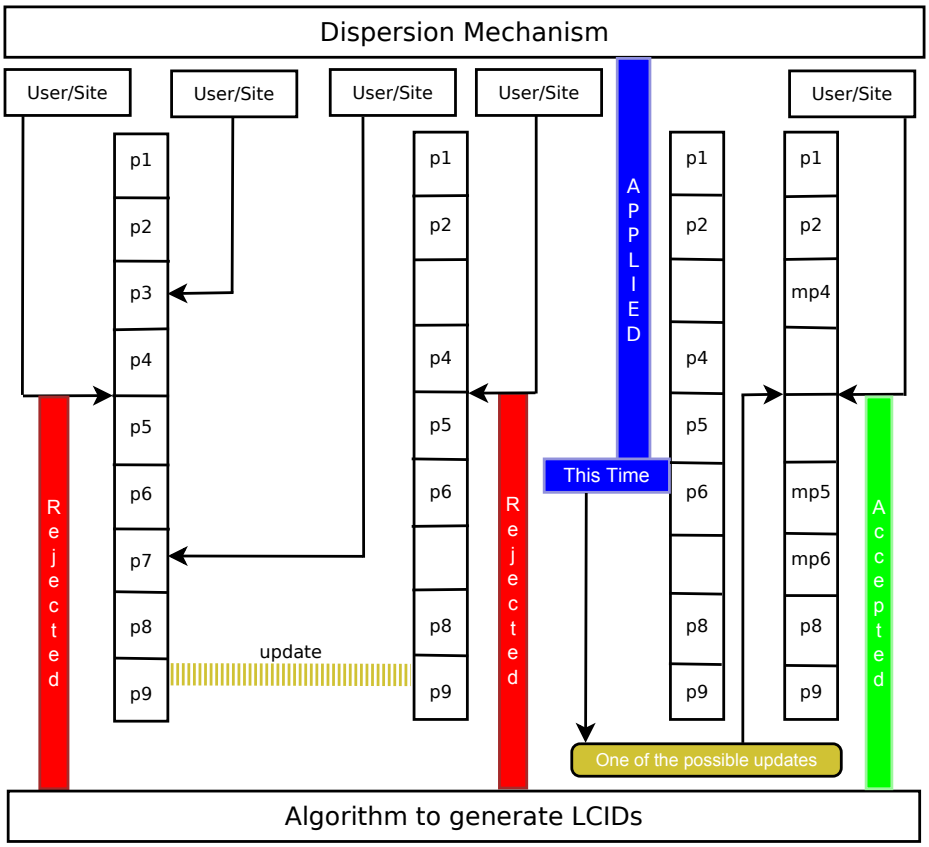


FIGURE 11.1 – Diagramme général pour le problème de dispersion

Ensuite, nous expliquons une des situations possibles et proposons un des algorithmes possibles qui pourraient être utiles pour comprendre le problème de dispersion.

11.1.2 Une des situations possibles

Supposons que il y a deux utilisateurs U_1 et U_2, assignée avec $USIDs$, ϵ_1 et ϵ_2 tel que $USID_{max} = max(\epsilon_1, \epsilon_2)$, calcule les ensembles S_1 et S_2 de $LCIDs$ et $S = S_1 \cup S_2$.
Soit p_j, $1 \leq j \leq Cd_S$ dénote position des points dans l'ensemble S, où Cd_S dénote nombre

des éléments dans l'ensemble S. Les utilisateurs U_1, U_2 sont pas autorisé à enlever des points à positions p_j, pour $j = 1$ et/ou $j = Cd_S$.
Soit n_r dénote nombre de points enlevés et n_{in} dénote le nombre de points que l'utilisateur peut insérer alors $n_{in} = n_r$.
Maintenant, let U_1/U_2 enlève n_r points consécutifs à la position p_j, $n_1 \leq j \leq n_2$ pour certains n_1, $n_2 \in \mathbb{N}$.

11.1.3 Cas-1

Soit U_1/U_2 veut insérer des points à la même position p_j, $n_1 \leq j \leq n_2$. Dans ce cas-là, en raison de la propriété de fermeture, les utilisateurs U_1/U_2 recalculera les points qui a été déjà enlevé par U_1/U_2.

11.1.4 Cas-2

Soit U_1/U_2 veut insérer les points à la position différent que p_j, $n_1 \leq j \leq n_2$, $n_1 > 1$, $n_2 < Cd_S$, puis utilisateurs U_1/U_2 insèreront les points $n_{in} = n_r$ à position p_j, pour $n_0 \leq j < n_1$, ou, $n_2 < j \leq n_3$, $n_2 > 1$, $n_3 < Cd_S$, $n_0 > 1$.

- Let p_j, $n_0 \leq j < n_1$, $n_0 > 1$ et $n_{in} = n_r$. Dans ce cas-là, nous avons deux options.
  1. Nous ajoutons le $\delta * n_r$ à tous les points au position p_j, $n_1 \leq j \leq n_2$ et poussons vers du côté droit à créer l'espace avant d'insérer le $ndepoints_rle$, où le $\delta = 1 \times 10^{-p_r}le$.
  2. Nous ajoutons le δ à tous les points au position p_j, $n_1 \leq j \leq n_2$ chaque fois avant d'insérer chaque point et les déplaçons vers du côté droit pour créer l'espace et exécuter ce processus récursivement.
- Soit p_j, $n_2 < j \leq n_3$, et $n_{in} = n_r$. Dans ce cas, nous le répétons tout l'une des deux options ci-dessus et soustrayez *delta* à partir des points à la position p_j, $n_1 \leq j \leq n_2$ à leur déplacement vers la gauche.

Nous généralisons cette procédure sous la forme d'un algorithme comme suit :
Soit Cd_SR représente cardinalité de l'ensemble S après avoir enlevé quelques points.

L'algorithme

1. Si $Cd_{SR} = Cd_S$, alors aucune insertion n'est possible.
2. Si $Cd_{SR} \neq Cd_S$, alors exécuter l'Algorithme 2
   (a) Si les conditions de l'algorithme 2 sont satisfaits, puis insérez le point / points, sinon passez à l'étape suivante
   (b) Soit utilisateur désirs d'insérer un point entre les positions p_j et P_{j+1}, alors localiser les positions vides dans les deux directions, à la gauche de p_j, et à droite de la point p_{j+1}.
   Supposons que p_{j+q_1} ou p_{j-q_2} est vide position pour certain q_1, $q_2 \in \mathbb{N}$ puis
   - Ajouter *delta* à point (points) à la position (position) p_i, pour, $j + 1 \leq i \leq j + q_1$, OU,
   - Sous-tract δ à point (points) à position (positions) p_i, pour, $j - q_2 \leq i \leq j$, OU,
   - On peut exécuter les deux opérations simultanément.

   (c) Insérez le point si elle est différente de s, $\forall s \in S$, où $S = \bigcup S_i$.
3. Continuent processus jusqu'à ce que condition $Cd_{SR} \neq Cd_S$ est valide.

11.1.5 Calcul séquentiel

En partie I (Chapitre 5), nous prouvons l'existence de *"In Situ Design of Computation (IDC)"* sur l'ensemble des nombres entiers. Comme un travail de plus, c'est encore restent enquêter sur l'existence d'un tel *"In Situ Design of Computation (IDC)"* pour les virgules flottantes. Étude des limites basées sur les nombres de Fibonacci attire des chercheurs de développer des relations plus qui vous aideront à trouver le nombre minimum de devoirs exigé. La réalisation de ce résultat pourrait être une conséquence possible une grande dans le contexte du résultat de Gabriel Lamé (mathématicien français) pour la limite supérieure de plus algorithme d'Euclide dû au fait que le *"In Situ Design of Computation (IDC)"* sur des entiers est une application indirecte de l'algorithme d'Euclide. L'identité 5.2.2 conduit à décomposer les matrices qui pourraient conduire à trouver un certain nombre d'applications en algèbre linéaire et de la zone pertinente. La décomposition polynomiale sur GF(2) pour *"In Situ Design of Computation (IDC)"* conduit à étudier plus avant l'existence de ce type de décomposition pour le cas général.

11.2 Conclusion

Cette thèse conduit à la conception décentralisée du système d'édition collaborative basée sur la méthode d'indexage de précision de contrôle introduits dans la partie II. Avec l'évaluation de quelques techniques existantes, il explorent des algorithmes, des applications et l'expérimentation exécutés pour vérifier les résultats. Une telle communication indexée entre le système distribué garantit la réduction de conflits. Comme l'unicité d'identificateurs pour le fait d'établir un index est prouvée, donc, il n'y a aucune chance de conflits dans l'échange ordonné ou au hasard de points. Un schéma prescrit des identifiants générés autorisés à les emballer dans l'espace, mis attribués ou une table, d'où, ils sont faciles d'accès. En parallèle, cette thèse introduit *"In Situ Design of Computation (IDC)"* dans le calcul séquentiel qui mène vers l'optimisation (processeur, compilateur, la mémoire, le code) avec une décomposition polynomiale sur GF (2). Ce style de décomposition attire communauté de la recherche pour complément d'enquête et de mise en œuvre pratique qui pourrait mener vers la conception de l'unité arithmétique.

Glossaire

Éléments : Éléments décrivent etc caractères, des lignes, des objets qui pourraient être modifiés au cours du processus d'édition collaborative.

DCE : DCE stands pour l'édition collaborative décentralisée, où les utilisateurs sont dispersés géographiquement.

DCE Modèle : Un modèle d'édition collaborative décentralisée basé sur la méthode d'indexage de précision contrôlée.

Dispersion Mécanisme : Une stratégie possible qui permet d'utiliser la mémoire existante de telle sorte que le maximum de cardinalité pourrait être obtenue.

First Tool : Une méthode pour construire polynôme sur GF(2) de telle sorte que l'IDC peuvent être construits pour les applications booléen.

IDC : IDC dénote In Situ Design of Computation, Un calcul séquentiel qui n'utilisent pas de variables supplémentaires autres que les variables disponibles en entrée.

LCID : LCID dénote les identifiants que nous assignons à des lignes ou caractères dans l'édition d'un document partagé en collaboration.

Local cardinalité : Le nombre possible d'identifiants $LCIDs$ qui pourrait être créé pour un utilisateur/site sous la précision particulière est appelé cardinalité local et est dénoté par C_l.

Notation ($\aleph$) : Notation $\aleph$ utilisé pour dénoter un réseau comporte n ($n \in \mathbb{N}$) utilisateurs, sites ou peers,

PCT : Technique de précision contrôlée qui permet de créer des identifiants uniques réel.

Points : Points décrire les identifiants ainsi que des éléments parce que l'échange d'éléments de provoquer l'échange des identifiants trop.

POSID : POSID dénote la position des identificateurs créés pour associer avec des personnages (lignes) dans l'édition collaborative.

Précision : nombre de chiffres après le point d'une valeur (arrondi à décimales / aux chiffres significatifs), *e.g.*, la précision de valeurs $'12,34600\ 12,345$ sont 5 et 3, respectivement.

Précision (p_ϵ) : est une notation pour la précision choisie pour créer des identifiants USID.

Round a value : Une fonction qui arrondit la valeur selon la définition de Précision.

USID : USID dénote les identifiants que nous assignons à des utilisateurs ou des sites participant à l'édition d'un document partagé en collaboration.

Index

Bibliographie

[1] D.P. Agrawal. Graph theoretical analysis and design of multistage interconnection networks. *Computers, IEEE Transactions on*, C-32(7) :637 –648, july 1983.

[2] Richard Beigel. The polynomial method in circuit complexity. In *Structure in Complexity Theory Conference*, pages 82–95, 1993.

[3] Jean Claude Bermond, Jean Michel Fourneau, and Alain Jean-Marie. A graph theoretical approach to equivalence of multistage interconnection networks. *Discrete Appl. Math.*, 22 :201–214, March 1989.

[4] Florent Bouchez, Alain Darte, Christophe Guillon, and Fabrice Rastello. Register Allocation : What does the NP-completeness Proof of Chaitin et al. Really Prove ? In *Fifth Annual Workshop on Duplicating, Deconstructing and Debunking (WDDD2006) (at ISCA-33)*, June 2006. `http://perso.ens-lyon.fr/fabrice.rastello/Biblio_Perso/Articles/WDDD06.pdf`.

[5] Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to graph coloring register allocation. *ACM Trans. Program. Lang. Syst.*, 16(3) :428–455, 1994.

[6] Serge Burckel. Closed Iterative Calculus. In *Theoretical Computer Science*, pages 371–378, 1996.

[7] Serge Burckel. The parallel-sequential duality : Matrices and graphs. *CoRR*, abs/0709.4397, 2007.

[8] Serge Burckel and E. Gioan. In situ design of register operations. In *Proceedings of ISVLSI'08*, pages 287–292, 2008.

[9] Serge Burckel, Emeric Gioan, and Emmanuel Thomé. Mapping computation with no memory. In *Proceedings of the 8th International Conference on Unconventional Computation*, UC '09, pages 85–97, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] Serge Burckel and M. Morillon. Three Generators for Minimal Writing-Space Computations. In *Theoretical Informatics and Application*, volume 34, pages 131–138, 2000.

[11] Serge Burckel and M. Morillon. Quadratic Sequential Computations of Boolean Mappings. In *Theory of Computing Systems. 37(4)*, pages 519–525, 2004.

[12] Serge Burckel and M. Morillon. Sequential computation of linear boolean mappings. In *Theoretical Computer Science, (314)*, pages 287–292. Springer, 2004.

[13] G. J. Chaitin. Register allocation & spilling via graph coloring. In *SIGPLAN '82 : Proceedings of the 1982 SIGPLAN symposium on Compiler construction*, pages 98–105, New York, NY, USA, 1982. ACM.

[14] Gregory J. Chaitin, Marc A. Auslander, Ashok K. Chandra, John Cocke, Martin E. Hopkins, and Peter W. Markstein. Register allocation via coloring. *Comput. Lang.*, 6(1) :47–57, 1981.

[15] Siddhartha Chatterjee, Vibhor V. Jain, Alvin R. Lebeck, Shyam Mundhra, and Mithuna Thottethodi. Nonlinear array layouts for hierarchical memory systems. In *Proceedings of the 1999 ACM International Conference on Supercomputing*, pages 444–453, 1999.

[16] A. Church. An unsolvable problem in elementary number theory. *American Journal of Mathematics*, 58 :356–363, 1936.

[17] Sandy Citro. *A Framework for Real Time Collaborative Editing in a Mobile Replicated Architectur*. PhD thesis, School of Computer Science and Information technology, RMIT University, Melbourne, Victoria, Australia, June 2007.

[18] Sandy Citro, Jim McGovern, and Caspar Ryan. Conflict management for real-time collaborative editing in mobile replicated architectures. In Gillian Dobbie, editor, *Thirtieth Australasian Computer Science Conference (ACSC2007)*, volume 62 of *CRPIT*, pages 115–124, Ballarat Australia, 2007. ACS.

[19] Y. Crama and Peter L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications.* In preparation, December 02, 2008.

[20] Clarence A. Ellis and Simon J. Gibbs. Concurrency Control in Groupware Systems. In *Proceedings of the ACM SIGMOD Conference on the Management of Data - SIGMOD'89*, pages 399–407, Portland, Oregon, USA, May 1989. ACM Press.

[21] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. Groupware : some issues and experiences. *Commun. ACM*, 34 :39–58, January 1991.

[22] Bogdan J. Falkowski. A note on the polynomial form of boolean functions and related topics. *IEEE Trans. Comput.*, 48 :860–864, August 1999.

[23] Matteo Frigo, Charles E. Leiserson, Harald Prokop, Sridhar Ramachandran, and Z W(l. Cache-oblivious algorithms (extended abstract). In *Proceedings of 40th Annual Symposium on Foundations of Computer Science*, pages 285–397. IEEE Computer Society Press, 1999.

[24] Lal George and Andrew W. Appel. Iterated register coalescing. *ACM Trans. Program. Lang. Syst.*, 18(3) :300–324, 1996.

[25] Saul Greenberg and David Marwood. Real time groupware as a distributed system : concurrency control and its effect on the interface. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 207–217, New York, NY, USA, 1994. ACM.

[26] K. Hoste and L. Eeckhout. Cole : Compiler optimization level exploration. In *the International Symposium on Code Generation and Optimization (CGO)*, 2008.

[27] Charles McLaughlin Hymes and Gary M. Olson. Unblocking brainstorming through the use of a simple group editor. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, CSCW '92, pages 99–106, New York, NY, USA, 1992. ACM.

[28] Claudia-Lavinia Ignat, Gérald Oster, Pascal Molli, Michèle Cart, Jean Ferrié, Anne-Marie Kermarrec, Pierre Sutra, Marc Shapiro, Lamia Benmouffok, Jean-Michel Busca, and Rachid Guerraoui. A comparison of optimistic approaches to collaborative editing of Wiki pages. In *Collaborative Comp. : Networking, Apps. and Worksharing (CollaborateCom)*, number 3, White Plains, NY, USA, November 2007.

[29] Abdessamad Imine. Coordination model for real-time collaborative editors. In John Field and Vasco Thudichum Vasconcelos, editors, *COORDINATION*, volume 5521 of *Lecture Notes in Computer Science*, pages 225–246. Springer, 2009.

[30] Abdessamad Imine, Pascal Molli, Gérald Oster, and Michaël Rusinowitch. Proving correctness of transformation functions in real-time groupware. In Kari Kuutti, Eija Helena Karsten, Geraldine Fitzpatrick, Paul Dourish, and Kjeld Schmidt, editors, *ECSCW*, pages 277–293. Springer, 2003.

[31] Abdessamad Imine, Michaël Rusinowitch, Gérald Oster, and Pascal Molli. Formal design and verification of operational transformation algorithms for copies convergence. *Theor. Comput. Sci.*, 351(2) :167–183, 2006.

[32] Michael Jurczyk. Performance comparison of wormhole-routing priority switch architectures. In *Proceedings of the international conference on parallel and distributed processing techniques and applications 2001 (PDPTA'01)*, pages 1834–1840, Las Vegas, USA, 2001.

[33] A. Karsenty and M. Beaudouin-Lafon. An algorithm for distributed groupware applications. In *Distributed Computing Systems, 1993., Proceedings of the 13th International Conference on*, pages 195–202, May 1993.

[34] J. Keller. The 21264 : a superscalar alpha processor with out-of-order execution. *9th Annual Microprocessor Forum*, 1996.

[35] S. C. Kleene. λ-definability and recursiveness. *Duke Mathematical Journal*, 2 :340–353, 1936.

[36] Michael Knister and Atul Prakash. Issues in the design of a toolkit for supporting multiple group editors. *Computing Systems – The Journal of the Usenix Association*, 6 :135–166, 1993.

[37] A. Kumar. The HP PA-8000 RISC CPU : A high performance out-of-order processor. In *Hot Chips VIII*, 1996.

[38] Monica S. Lam, Edward E. Rothberg, and Michael E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 63–74, 1991.

[39] Gabriel Lamé. Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers. volume 19, pages 867–870, Paris, 1844. Comptes Rendus de l'Académie des sciences.

[40] Leslie Lamport. Time, Clocks, and the Ordering of events in a Distributed System. *Communications of the ACM*, 21(7) :558–565, July 1978.

[41] Stefania Leone, Thomas B. Hodel-Widmer, Michael H. Böhlen, and Klaus R. Dittrich. Tendax, a collaborative database-based real-time editor system. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *EDBT*, volume 3896 of *Lecture Notes in Computer Science*, pages 1135–1138. Springer, 2006.

[42] V. K. Leont'ev. Certain problems associated with boolean polynomials. *Computing Center, Russian academy of Sciences, ul. Vavilova 40, Moscow, GSP-1, 117967 Russia*, September 11, 1998.

[43] Du Li and Rui Li. Ensuring content and intention consistency in real-time group editors. In *ICDCS*, pages 748–755. IEEE Computer Society, 2004.

[44] Du Li and Rui Li. Preserving operation effects relation in group editors. In James D. Herbsleb and Gary M. Olson, editors, *CSCW*, pages 457–466. ACM, 2004.

[45] Rui Li and Du Li. A new operational transformation framework for real-time group editors. *IEEE Trans. Parallel Distrib. Syst.*, 18 :307–319, March 2007.

[46] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes / F.J. MacWilliams, N.J.A. Sloane.* North-Holland Pub. Co. ; sole distributors for the U.S.A. and Canada, Elsevier/North-Holland, Amsterdam ; New York : New York :, 1977.

[47] L. McGuffin and G. Olson. Shredit : A shared electronic workspace. Technical report, Cognitive Science and Machine Intelligence Laboratory, University of Michigan, 1992.

[48] Joon-Sang Park Michael, Michael Penner, and Viktor K Prasanna. Optimizing graph algorithms for improved cache performance. In *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS 2002), Fort Lauderdale, FL*, pages 769–782, 2002.

[49] John Miranda. The Usage of Compiler Optimization by Programmers. In *Thesis, School of Engineering and Applied Science University of Virginia*, 2001.

[50] Judith S. Olson, Gary M. Olson, Marianne Storrøsten, and Mark Carter. How a group-editor changes the character of a design meeting as well as its outcome. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, CSCW '92, pages 91–98, New York, NY, USA, 1992. ACM.

[51] Gérald Oster. Tombstone transformation functions for ensuring consistency in collaborative editing systems. In *Proceedings of the 2nd International Conference on Collaborative Computing : Networking, Applications and Worksharing*. IEEE Press, 2006.

[52] Gérald Oster, Pascal Molli, Pascal Urso, and Abdessamad Imine. Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems. In *IEEE Conference on Collaborative Computing : Networking, Applications and Worksharing - CollaborateCom 2006 Collaborative Computing : Networking, Applications and Worksharing, 2006. CollaborateCom 2006. International Conference on*, pages 1–10, Atlanta, Georgia, USA, 11 2006. IEEE. `http://ieeexplore.ieee.org/`.

[53] Gérald Oster, Pascal Urso, Pascal Molli, and Abdessamad Imine. Proving correctness of transformation functions in collaborative editing systems. Research Report RR-5795, INRIA, 2005.

[54] Gérald Oster, Pascal Urso, Pascal Molli, and Abdessamad Imine. Data consistency for p2p collaborative editing. In Pamela J. Hinds and David Martin, editors, *CSCW*, pages 259–268. ACM, 2006.

[55] Neungsoo Park, Dongsoo Kang, Kiran Bondalapati, and Viktor K. Prasanna. Dynamic data layouts for cache-conscious factorization of dft. In *Proc. of International Parallel and Distributed Processing Symposium*, pages 693–701, 2000.

[56] Nuno Preguiça, Joan Manuel Marquès, Marc Shapiro, and Mihai Leţia. A commutative replicated data type for cooperative editing. *Distributed Computing Systems, International Conference on*, 0 :395–403, 2009.

[57] Wu Qinyi and Pu Calton. Consistency in real-time collaborative editing systems based on partial persistent sequences. Technical Report CERCS ; GIT-CERCS-09-07, Georgia Institute of Technology, 2009.

[58] C. P. Ravikumar, R. Aggarwal, and C. Sharma. A graph-theoretic approach for register file based synthesis. In *Proceedings of the Tenth International Conference on VLSI Design : VLSI in Multimedia Applications*, VLSID '97, pages 118–, Washington, DC, USA, 1997. IEEE Computer Society.

[59] Matthias Ressel, Doris Nitsche-Ruhland, and Rul Gunzenh äuser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. In *CSCW*, pages 288–297, 1996.

[60] Matthias Ressel, Doris Nitsche-Ruhland, and Rul Gunzenh äuser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, CSCW '96, pages 288–297, New York, NY, USA, 1996. ACM.

[61] Matthias Ressel, Doris Nitsche-Ruhland, Rul Gunzenhäuser, and Rul Gunzenh Auser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. pages 288–297. ACM Press, 1996.

[62] S. Rudeanu. On the decomposition of boolean functions via boolean equations. *j-jucs*, 10(9) :1294–1301, 2004. `http://www.jucs.org/jucs_10_9/on_the_decomposition_of`.

[63] K. S. McKinley S. Carr and C. Tseng. Compiler optimization for improving data locality. *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems.*

[64] Vivek Sarkar. Code optimization of parallel programs : evolutionary vs. revolutionary approaches. In *CGO '08 : Proceedings of the sixth annual IEEE/ACM international symposium on Code generation and optimization*, pages 1–1, New York, NY, USA, 2008. ACM.

[65] Vivek Sarkar. Challenges in code optimization of parallel programs. In *CC*, page 1, 2009.

[66] W.G. Schneeweiss. On the polynomial form of boolean functions : derivations and applications. *Computers, IEEE Transactions on*, 47(2) :217 –221, feb 1998.

[67] D. S. Scott and C. Strachey. Towards a mathematical semantics for computer languages. 1971.

[68] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Technical Report 7506, January 2011. `http://hal.archives-ouvertes.fr/inria-00555588/`.

[69] Chengzheng Sun and Clarence Ellis. Operational transformation in real-time group editors : issues, algorithms, and achievements. In *CSCW '98 : Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 59–68, New York, NY, USA, 1998. ACM.

[70] Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang, and David Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Trans. Comput.-Hum. Interact.*, 5 :63–108, March 1998.

[71] Chengzheng Sun and P. Maheshwari. An efficient distributed single-phase protocol for total and causal ordering of group operations. In *Proceedings of the Third International Conference on High-Performance Computing (HiPC '96)*, HIPC '96, page 295, Washington, DC, USA, 1996. IEEE Computer Society.

[72] Chengzheng Sun, Steven Xia, David Sun, David Chen, Haifeng Shen, and Wentong Cai. Transparent adaptation of single-user applications for multi-user real-time collaboration. *ACM Trans. Comput.-Hum. Interact.*, 13(4) :531–582, December 2006.

[73] David Sun and Chengzheng Sun. Operation context and context-based operational transformation. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06, pages 279–288, New York, NY, USA, 2006. ACM.

[74] R. M Tomasulo. An efficient algorithm for exploiting multiple arithmetic units. *IBM Journal of Research and Development*, pages 25–233, 1967.

[75] A. Turing. Computability and λ-definability. *Journal of Symbolic Logic*, 2 :153–163, 1937.

[76] Jonathan S. Turner and Riccardo Melen. Multirate clos networks. *IEEE Communications Magazine*, 41 :38–44, 2003.

[77] Dietmar Tutsch and Guenter Hommel. Performance of buffered multistage interconnection networks in case of packet multicasting. In *Proceedings of the 1997 Advances in Parallel and Distributed Computing Conference (APDC '97)*, APDC '97, pages 50–, Washington, DC, USA, 1997. IEEE Computer Society.

[78] S. F. Vinokurov and N. A. Peryazev. A polynomial decomposition of boolean functions. *Mathematical Notes*, 53 :130–133, 1993. 10.1007/BF01208315.

[79] Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot : A scalable optimistic replication algorithm for collaborative editing on p2p networks. In *ICDCS*, pages 404–412. IEEE Computer Society, 2009.

[80] Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot-undo : Distributed collaborative editing system on p2p networks. *IEEE Transactions on Parallel and Distributed Systems*, 21 :1162–1174, 2010.

[81] R. Clint Whaley and Jack J. Dongarra. Automatically tuned linear algebra software. In *Supercomputing '98 : Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–27, Washington, DC, USA, 1998. IEEE Computer Society.

[82] Tilman Wolf and Jonathan S. Turner. Design issues for high performance active routers. In *Proc. of the International Zurich Seminar on Broadband Communications*, pages 199–205, Zurich, Switzerland, February 2000.

[83] Yuanyuan Yang and Jianchao Wang. A class of multistage conference switching networks for group communication. In *Proceedings of the 2002 International Conference on Parallel Processing*, ICPP '02, pages 73–, Washington, DC, USA, 2002. IEEE Computer Society.

[84] B. Zhou and M. Atiquzzaman. A performance comparison of four buffering schemes for multistage interconnection networks, 2002.

Résumé

Dans cette thèse, nous montrons les capacités de calcul séquentiel et des systèmes distribués en étudiant *"In Situ Design of Computation (IDC)"* et en développant des méthodes pour faciliter la communication dans les systèmes décentralisés d'édition collaboratif.

La croissance rapide de l'utilisation de la technologie informatique moderne augmente la demande de meilleures performances dans tous les domaines de l'informatique. Cette demande pour des performances toujours plus grandes a conduit à la croissance des performances matérielles et à l'évolution de l'architecture, ce qui entraîne des contraintes sur la technologie de compilateur et pour la communauté de recherche. Étant donné que les microprocesseurs haute performance sont au cœur de chaque ordinateur à usage général, depuis les serveurs, les PC de bureau et portables, jusqu'aux téléphones cellulaires comme l'iPhone, l'augmentation du rendement est considéré comme un défi permanent dans l'informatique. En informatique, pour le calcul, les opérations élémentaires sont effectuées sur des objets élémentaires, *e.g.*, un processeur 32 bits ne peut effectuer des opérations que sur 32 bits. Ainsi, toute transformation de la structure de données doit être décomposée en opérations successives sur 32 bits. Pour échanger le contenu des registres, la solution habituelle pour assurer la complétude du calcul est de faire des copies à partir des données initiales. Mais cette solution peut générer des erreurs de mémoire lorsque les structures sont trop grandes, ou du moins diminue les performances de calculs. En effet, ces opérations impliquant plusieurs registres dans un micro-processeur, comme un compilateur ou un circuit électronique, nécessitent de copier certains registres dans la mémoire cache ou en mémoire vive, avec une perte de vitesse, ou de dupliquer les signaux de la puce elle-même, avec une consommation électrique accrue [9]. Pour faire face à ce problème, Burckel et al. [8, 9] étudient et étendent une stratégie qui effectue des opérations de telle manière qu'il ne nécessite aucune variable supplémentaire autres que les variables disponibles en entrée. Cette thèse contribue à ce type de stratégies, en vérifiant les résultats actuels à travers l'expérimentation et l'illustration, en étudiant et en cherchant de nouvelles orientations telles que l'idée sous-jacente pourrait être étendue et mise en œuvre avec succès.

En contribuant à l'étude des systèmes distribués, cette thèse pose les bases pour la conception de systèmes décentralisés d'édition collaborative, en introduisant une méthode d'indexation de contrôle de précision. En fait, les Realtime Collaborative Editors (RCE) permettent à des groupes d'utilisateurs de modifier simultanément le document partagé à partir de sites physiquement distants interconnectés par un réseau informatique. Dans de tels systèmes distribués, l'un des défis est d'éviter des conflits de communication. A cet effet, la communication indexée semble être une exigence essentielle. Dans cette thèse, une méthode d'indexation de contrôle de précision est introduite pour générer des identifiants uniques, qui sont utilisés pour la communication indexée dans les systèmes distribués, en particulier dans les systèmes décentralisés d'édition collaborative. Ces identifiants sont toujours des nombres réels avec un modèle de précision spécifique et contrôlé. L'ensemble d'identifiants est maintenu fini, ce qui permet de calculer des cardinalités locales et globales. Cette propriété joue un rôle important dans le traitement de la communication indexée. Outre cela, certaines autres propriétés, dont la préservation de l'ordre sont observées. La méthode d'indexation est testée et vérifiée par l'expérimentation avec succès et elle conduit à la conception

du système décentralisé d'édition collaborative.

Mots-clés: système d'édition collaborative, modèle de cohérence des données, structure de données persistante, P2P, CRDT, réplication optimiste, algorithmes, optimisation de la mémoire, processeur d'optimisation, la logique et de calcul, la conception de circuits