2013-2014
Master 2 Informatique
Intelligence décisionnelle

Caractérisation de données biologique

Minimisation du problème de caractérisation multiple

Jean-Mathieu CHANTREIN

Sous la direction de Messieurs
Frédéric LARDEUX
Frédéric SAUBION

Soutenu publiquement le :
Vendredi 04 Juillet 2014

université angers

ENGAGEMENT DE NON PLAGIAT

Je, soussigné Jean-Mathieu CHANTREIN déclare être pleinement conscient(e) que le plagiat de documents ou d'une partie d'un document publiée sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.
En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour écrire ce rapport ou mémoire.

Angers, le 10 / 04 / 2014

Présidence de l'université
40 rue de rennes – BP 73532
49035 Angers cedex
Tél. 02 41 96 23 23 | Fax 02 41 96 23 00

université angers

Introduction

Sujet du stage

La plupart des bactéries appartenant au genre Xanthomonas sont responsables de pathologies sur une large gamme de cultures économiquement importantes, induisant notamment des pertes de rendement et diminuant ainsi la valeur marchande des semences. Quelques graines contaminées suffisent à générer une source d'inoculation primaire et à occasionner ainsi une dissémination ultérieure plus large. En particulier, le pathovar phaseoli de Xanthomonas axonopodis (Xap) qui regroupe toutes les souches identifiées comme pathogènes sur le haricot n'est pas endémique en Europe mais pour limiter son introduction, il est inscrit sur la liste des agents pathogènes de quarantaine. Une approche possible pour l'identification des souches bactériennes consiste à utiliser un répertoire de gènes de virulence. Il s'agit ainsi de trouver la plus petite combinaison de gènes de virulence spécifiques. Cette combinaison peut ainsi être utilisée pour concevoir un test d'identification. Des travaux préliminaires[CLSZ13] montrent que la combinaison des tests moléculaires ainsi obtenus fournit une technique rapide pour l'identification de toutes les souches de Xanthomonas pathogènes sur les haricots.

Avec les possibilités accrues d'acquisition de données génomiques – par exemple le séquençage à haut débit – mais également phénotypiques, le problème de la caractérisation de données biologiques devrait rapidement devenir l'un des verrous essentiel de l'exploitation effective des grandes bases de données qui sont en cours de constitution, et constituera donc un centre d'intérêt commun aux biologistes des domaines du végétal ou de la santé. La caractérisation telle que nous l'entendons permet d'identifier les caractères propres, éventuellement hétérogènes, d'un groupe d'individus partageant des spécificités fonctionnelles communes (par exemple pathologiques).

Du point de vue informatique, ce problème est abordé comme la recherche d'un ensemble de formules propositionnelles (variables booléennes) permettant de caractériser de manière exacte les groupes de pathogènes. Les algorithmes mis en jeu reposent sur des explorations arborescentes (Branch & Bound) et des algorithmes heuristiques (recherche locale).

L'objectif de ce stage est double :

— D'une part il s'agit de constituer de nouveaux jeux de données, en dialogue avec nos collègues biologistes. Ceci requiert la définition de formats et l'exploration de base de données pour la collecte d'informations pertinentes. Ce travail sera effectué en lien étroit avec les laboratoires de l'INRA Angers.

— D'autre part, il s'agit également d'améliorer les algorithmes existants et de proposer de nouvelles approches pour traiter des instances de grande taille. Cette phase s'inspire des algorithmes de résolution de problèmes combinatoires (SAT-CSP).

Plan du rapport

Dans la première partie de ce mémoire, nous dressons un état de l'art concernant l'étude du MIN-PCM. Celui-ci reprend les grands axes présentés dans l'article [CLSZ13]. Ensuite nous décrivons les contributions que nous apportons à l'étude de ce problème : nous définissons la notion d'instance difficile, nous proposons des heuristiques de résolution (exactes ou approchées) et nous fournissons les résultats ainsi obtenus. Enfin, nous tirons les conclusions de nos travaux et nous discutons les perspectives de recherche envisagées.

Table des matières

1 État de l'art

1.1 Présentation générale du problème

La plupart des bactéries appartenant au genre Xanthomonas sont responsables de pathologies sur une large gamme de cultures économiquement importantes, induisant notamment des pertes de rendement et diminuant ainsi la valeur marchande des semences. Quelques graines contaminées sont suffisantes pour générer une source d'inoculation primaire et occasionner ainsi une dissémination ultérieure plus large. En particulier, le pathovar [1] phaseoli de Xanthomonas Axonopodis (Xap) qui regroupe toutes les souches identifiées comme pathogènes sur le haricot [VHKS95], n'est pas endémique en Europe mais pour limiter son introduction, il est inscrit sur la liste des agents pathogènes de quarantaine.

La taxonomie du genre Xanthomonas n'est pas encore pleinement résolue, et la délimitation de certaines espèces dans ce genre fait encore débat [SPL+05] ; les approches phylogénétiques ne sont alors pas réellement applicables. Une approche possible pour l'identification des souches bactériennes consiste à utiliser un répertoire de gènes de virulence. Il s'agit ainsi de trouver la plus petite combinaison de gènes de virulence spécifiques au Xap. Cette combinaison peut être utilisée pour concevoir un test PCR multiplex pour l'identification de Xap [BKC+13, BCH+12] dont le coût est directement lié au nombre de gènes à tester. Les résultats obtenus montrent que la combinaison des tests moléculaires ainsi obtenus fournit une technique rapide pour l'identification de toutes les souches de Xanthomonas pathogènes sur les haricots.

Plus formellement, considérons un ensemble d'entités (les souches bactériennes) regroupées en groupes (les pathovars). Chaque entité est définie par la présence ou l'absence d'un ensemble de caractères (les gènes). Au regard de la représentation binaire qui est utilisée, une entité est considérée comme une interprétation booléenne sur les caractères, qui seront donc les variables booléennes du problème. Ainsi, pour chaque groupe, l'ensemble des entités fournit une table de vérité partielle d'une fonction booléenne vraie pour les interprétations correspondant aux entités du groupe et fausse pour toutes les autres entités des autres groupes. Une telle fonction sera appelée caractérisation d'un groupe. Ces fonctions booléennes seront représentées par des formules propositionnelles construites sur un langage fixé.

Le problème de caractérisation multiple consiste ainsi à trouver un ensemble de formules booléennes de sorte que chaque formule soit une caractérisation. Le terme multiple désigne le fait qu'il faut considéroer un ensemble de groupes dont les caractérisations sont dépendantes des entités contenues dans ceux-ci mais aussi des entités appartenant aux autres groupes.

Souches	Groupes	Caractères		
		a	b	c
e_1	g_1	0	0	0
e_2		0	0	1
e_3	g_2	1	1	1
e_4	g_3	1	1	0
e_5		0	1	0

FIGURE 1 – Exemple de problème de caractérisation multiple

Dans la figure 1 il y a 5 entités réparties dans 3 groupes et dont la description se base sur un ensemble de 4 caractères. Résoudre ce problème revient à caractériser chaque groupe. Il faut donc, pour chaque groupe, trouver une combinaison de variables permettant de construire une formule vraie pour toutes les entités du groupe et fausse pour les autres entités des autres groupes. Dans l'exemple de la figure 1, le groupe 1 est caractérisé par la négation des variables a et b alors que le groupe 2 est caractérisé par les variables b et c. Les souches du groupe 3 ont toutes en commun la négation de la variable c tout comme l'entité e_1 du groupe 1. Il faut donc ajouter une autre variable (b par exemple) pour être sûr de caractériser le groupe.

1. La notion de pathovar correspond à une subdivision du genre ayant des caractéristiques pathologiques observées communes.

1.2 Problème de caractérisation multiple

Nous présentons dans cette section le problème de caractérisation multiple ainsi que les diverses méthodes qui ont été proposés dans [CGLS12, CLSZ13] et qui permettent de résoudre le MIN-PCM.

1.2.1 Présentation du problème

Définition 1 (Instance du PCM) *Une instance du problème de caractérisation multiple est définie par un n-uplet $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ où $\mathcal{X}$ est l'ensemble des variables propositionnelles, $\mathcal{E}$ est l'ensemble des entités définies sur $\mathcal{X}$ et $\mathcal{G} \subseteq 2^{\mathcal{E}}$.*

Chaque entité représente une affectation booléenne, ou interprétation, définit ainsi $e : \mathcal{X} \to \{0, 1\}$, où 0 et 1 sont respectivement les valeurs de vérité fausse et vraie. $e(x)$ correspond donc à la valeur de vérité affectée à x dans l'interprétation e. Pour une formule propositionnelle ϕ quelconque sur $\mathcal{X}$, nous notons $e \models \phi$ le fait que l'interprétation $e \in \mathcal{E}$ satisfait la formule ϕ. Une entité $e \in \mathcal{E}$ sera classiquement représentée par un n-uplet de valeur booléennes (un élément de $\{0, 1\}^n$). Ainsi, une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ peut être vue comme une matrice booléenne dont les n colonnes correspondent aux variables booléennes de $\mathcal{X}$ et les m lignes aux entités de $\mathcal{E}$. Chaque variable x_j correspond à la colonne $j \in \{1, \ldots, n\}$. Chaque entité e_i correspond à une ligne $i \in \{1, \ldots, m\}$.

Nous pouvons appliquer des prétraitements pour réduire la taille de la matrice en réduisant le nombre de variables réellement utiles et/ou le nombre d'entités. Nous définissons ainsi la notion d'instance non redondante.

Définition 2 (Instance non redondante) *Une instance est non redondante ssi :*

- *Il n'existe pas de colonnes dont toutes valeurs sont identiques :* $\nexists j \in \{1, \ldots, n\}, \forall i \in \{1, \ldots, m\}$ *tel que* $a_{ij} = 1$ *(resp* $a_{ij} = 0$*);*
- *Chaque colonne est unique :* $\nexists j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, n\} \setminus \{j\}, \forall i \in \{1, \ldots, m\}$ *tel que* $a_{ij} = a_{ik}$ *;*
- *Chaque entité est unique :* $\nexists i \in \{1, \ldots, m\}, \forall l \in \{1, \ldots, m\} \setminus \{i\}, \forall j \in \{1, \ldots, n\}$, *tel que* $e_i(x_j) = e_l(x_j)$.

L'application d'un algorithme d'élimination de la redondance s'effectue au pire des cas en $\mathcal{O}(|\mathcal{X}|^2+|\mathcal{E}|^2)$ et peut conduire à une instance mal-formée dont toutes les colonnes ont été supprimées ou possédant des groupes vides.

Une fois l'instance réduite, nous pouvons préciser la notion de solution d'un problème de caractérisation multiple.

Définition 3 (Caractérisation d'un groupe) *Pour une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$, une formule ϕ_g caractérise un groupe $g \in \mathcal{G}$ ssi :* $\forall e \in g, e \models \phi_g$ *(acceptation des entités du groupe) et* $\forall g' \in \mathcal{G} \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g$ *(rejet des entités des autres groupes).*

Par extension, nous notons $g \models \phi_g$ le fait que ϕ_g caractérise g selon la définition précédente. $Sol(g)$ représente l'ensemble des solutions d'un groupe g. $Sol(g) = \{\phi_g | g \models \phi_g\}$.

Définition 4 (Solution d'un PCM) *Pour une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$, une solution admissible du PCM est un $|\mathcal{G}|$-uplet de formules $\Phi = (\phi_1, \cdots, \phi_{\mathcal{G}})$ tel que $\forall i \in \{1, \ldots, |G|\}, g_i \in \mathcal{G}$, $g_i \models \phi_i$.*

Soit $\mathcal{I} = (\mathcal{X}, \mathcal{E}, \mathcal{G})$, $SOL(\mathcal{I})$ est l'ensemble de toutes les solutions multiples pour tous les groupes. $SOL(\mathcal{I}) = Sol(g_1) \times \cdots \times Sol(g_{|\mathcal{G}|})$, où $\times$ désigne le produit cartésien. Pour un ensemble de groupes $\mathcal{G}$ et un $|G|$-uplet de formules $\Phi = (\phi_1, \cdots, \phi_{|\mathcal{G}|})$, nous notons par extension $\mathcal{G} \models \Phi$ le fait que $\forall i \in \{1, \ldots, |G|\}, g_i \in \mathcal{G}$, $g_i \models \phi_i$.

Définition 5 (Satisfiabilité d'un PCM) *Une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ est satisfiable (resp. insatisfiable) ssi* $\forall g \in \mathcal{G}, Sol(g) \neq \emptyset$ *(resp.* $\exists g \in \mathcal{G}, Sol(g) = \emptyset$*).*

Il est évident que toute instance possédant deux interprétations identiques dans deux groupes différents n'est pas satisfiable d'où :

Proposition 1 *Une instance $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ est satisfiable* $\Leftrightarrow \forall g, g' \in \mathcal{G}, g \neq g', g \cap g' = \emptyset$.

Une conséquence immédiate est qu'une instance non redondante et bien formée est satisfiable puisque chaque entité a une interprétation unique. En pratique, le test de satisfiabilité est effectué implicitement lors de l'étape de prétraitement et à charge de l'utilisateur de modifier ou supprimer les entités mis en cause dans l'échec de ce test.

Définition 6 (Taille d'un n-uplet de formule) *Pour un n-uplet* $\Phi = (\phi_1, \cdots, \phi_n)$ *nous avons* $|\Phi| = |\bigcup_{\phi_i} var(\phi_i)|$*, où* $var(\phi)$ *retourne l'ensemble des variables de* ϕ*.*

Définition 7 (k-PCM (problème de décision)) *Soit une instance* $\mathcal{I}$ *une caractérisation multiple minimale* k *est un ensemble de formules* $\Phi \in Sol(\mathcal{I})$ *vérifiant* $|\Phi| \leq k$ *avec* $k \in \mathbb{N}^+$*.*

Le problème de caractérisation multiple minimale pour une taille k ne correspond pas nécessairement à une solution minimale de $(\phi_1, \cdots, \phi_n)$ telle que chaque ϕ_i est un élément minimal de $Sol(g_i)$. Nous définissons le problème d'optimisation comme suit.

Définition 8 (MIN-PCM (problème d'optimisation)) *Pour une instance* $\mathcal{I}$*, une caractérisation optimale multiple minimale est un ensemble de formules* $\Phi^* \in Sol(\mathcal{I})$ *vérifiant* $|\Phi^*| \leq |\Phi|$ *avec* $\forall \Phi \in Sol(\mathcal{I})$

Minimisation du problème de caractérisation multiple

La minimisation du problème de caractérisation multiple consiste à définir le plus petit nombre de gène pouvant caractériser une instance PCM.

1.3 Complexité

Le problème SET-COVER appartient à la classe de complexité W[2]-complet. Il a été montré dans [CLSZ13] qu'une instance PCM pouvait être réduite en temps polynômial en une instance SET-COVER. Il en résulte que PCM appartient à la classe de complexité W[2]-complet [2]. Dès lors, il a été prouvé que le MIN-PCM appartient à la classe de complexité W[2]-difficile. L'impact direct de l'appartenance de MIN-PCM à cette classe de complexité est que l'**unique** possibilité d'améliorier significativement la résolution complète [3] d'une instance est de **travailler sur des heuristiques de choix de variables**(gènes).

1.4 Méthode de résolution

Toutes les méthodes présentées dans cette sous-section sont issus de [CLSZ13].

1.4.1 Une résolution basés sur les fonctions booléenes partiellement définis

Les fonctions booléennes partiellement définies, *partially defined Boolean formula - pdBf*, [MiHI99] permettent de proposer un cadre de résolution intéressant.

Une pdBf est vue comme une fonction booléenne pour laquelle certaines interprétations ne sont pas définies. La classe C^+ (resp. C^-) désigne l'ensemble des exemples positifs (resp. négatifs). À partir de toute fonction booléenne, nous pouvons calculer une formule caractérisant l'ensemble des interprétations, appelée extension. Il en est de même pour les pdBf où une DNF (formule en forme normale disjonctive) est une extension facilement calculable caractérisant la classe $\mathcal{C}^+$.

En ce qui concerne le PCM, nous construisons un ensemble de pdBf emboîtées où chaque classe C^- est l'union des classes C^+ des autres groupes. Nous nous appuyons sur la notion de projection pour calculer de nouvelles solutions.

Définition 9 (Projection) *Une projection* π *d'une instance* $\mathcal{I}$ *de PCM est la donnée d'un sous-ensemble* $\mathcal{X}' \subseteq \mathcal{X}$*, définissant implicitement l'instance* $\mathcal{I}' = (\mathcal{X}', \{\pi(e) \mid e \in \mathcal{E}\}, \{\pi(g) \mid g \in \mathcal{G}\})$*, où* $\pi(e)$ *est la restriction de* e *aux variables de* $\mathcal{X}'$*, et* $\pi(g)$ *est* $\{\pi(e) \mid e \in g\}$*. On appelle* dimension *d'une projection* π *le cardinal de* $\mathcal{X}'$*.*

La minimisation du nombre de colonnes pour le problème PCM revient à chercher la projection satisfiable de plus petite dimension associée aux pdBf de chaque groupe.

La principale difficulté du PCM ne dépend pas de la structure des formules de l'ensemble solution mais du choix des variables présentes dans celui-ci. La satisfiabilité d'un ensemble de formules de taille k pour une instance du PCM est équivalente au fait que nous cherchons un sous-ensemble de pdBf consistantes, issues d'une projection satisfiable de dimension k sur les pdBf initiales.

2. En admettant l'hypothèse que la WEFT-hiérarchie proposé par [Downey, Fellows, 1995] soit correcte.
3. Recherche exacte permettant de prouver l'optimalité d'une solution.

1.4.2 Résolutions complètes

Exact-Proj-Car Le solveur *Exact-Proj-Car*, proposé dans [CLGS12], est basé sur une approche complète consistant à valider ou non la présence d'une caractérisation de taille k. Deux types d'exploration sont possibles :

- Une exploration en largeur consistant à montrer qu'il n'existe aucune caractérisation valide de taille k avant de tester celles de taille $k+1$. En commençant avec $k = \lceil log_2(|\mathcal{G}|) \rceil$ il est donc garanti de trouver la caractérisation valide optimale. En effet, il est impossible de distinguer plus de 2^k classes avec une projection de dimension k.
- Une exploration en profondeur partant de la caractérisation maximale (toutes les variables) et recherchant une caractérisation valide de taille $k-1$ dès qu'une caractérisation valide de taille k est trouvée.

Ces deux approches garantissent toutes les deux de trouver une caractérisation valide optimale mais en cas d'arrêt de la recherche (temps limite atteint, ...), seule l'exploration en profondeur est capable de fournir une caractérisation valide. De plus, en termes de nombre de caractérisations testées, l'exploration en largeur semble la plus coûteuse (hormis pour les solutions de très petite taille). En effet, il est évident que l'exploration en largeur traitera au moins $\binom{n}{k}$ caractérisations (n étant le nombre de variables totales) alors que pour l'exploration en profondeur il est impossible de prévoir ce nombre (au mieux $n-k$ et au pire $\sum_{i=n}^{i=k} \binom{n}{i}$). Remarquons que l'utilisation conjointe de ces deux méthodes permet de borner la caractérisation optimale.

Ces explorations ont toutes les deux des choix de variables à faire : une heuristique de branchement basée sur un classement des variables de manière statique au début de la recherche. L'ordre utilisé est un calcul d'entropie inspirée par la technique proposée dans [DV81] qui privilégie les variables permettant de séparer un groupe par rapport aux autres.

Reformulation en programmation linéaire Il existe une reformulation du PCM en programmation linéaire. Nous nous intéressons plus particulièrement à la minimisation du PCM (MIN-PCM) qui consiste à minimiser la taille k de la solution. Cette reformulation permet d'obtenir de nouveaux résultats de complexité pour le PCM.

La modélisation du MIN-PCM peut se faire en programmation entière 0/1 (pseudo-booléen). Le MIN-PCM est reformulé sous la forme d'un problème MIN-ONES. Le problème MIN-ONES est un problème d'optimisation défini comme suit :

Définition 10 (MIN-ONES) *Soit Φ une collection de formules booléennes ϕ_i (contraintes) définie sur $\mathcal{X}$. Le problème consiste à trouver une affectation booléenne sur $\mathcal{X}$ telle que chaque contrainte soit satisfaite tout en minimisant le nombre de variable vraie dans cette affectation.*

Une formule booléenne est construite avec chaque paire d'entités de groupes différents.

Soit $(\mathcal{X}, \mathcal{E}, \mathcal{G})$ une instance du PCM. Pour toute paire d'entités $\{e_i, e_j\} \in \mathcal{E}^2$ telle que $e \in g, e' \in g', g \neq g'$, nous construisons une formule ϕ de la manière suivante :
$\phi = \bigvee_{x_k \in \{x | x \in \mathcal{X}, e(x) \neq e'(x)\}} x_k$.

Nous observons que ϕ est une clause composée uniquement de littéraux positifs. L'ensemble Φ de ces formules permet de modéliser entièrement le PCM de manière à le résoudre par programmation entière 0/1.

$$
\begin{array}{l}
Domaine : y_i \in \{0,1\} \\
min : \sum_i y_i \\
s.t. \\
y_1 + \cdots + y_n \geq 1, \forall (x_1 \vee \cdots \vee x_n) \in \Phi
\end{array}
$$

Nous constatons que la transformation d'une instance est bornée par $|\mathcal{E}|^2$

1.4.3 Résolution par recherche locale

LS-Proj-Car Utiliser une approche complète peut s'avérer inefficace dans le cas de PCM de grande taille. Une alternative est donc de résoudre le problème à l'aide d'un algorithme de recherche locale [HS04]. L'optimalité des résultats n'est plus garantie mais une solution de bonne qualité est généralement trouvée assez rapidement.

Le principe de l'approche appelée *LS-Proj-Car* est de parcourir l'espace des projections valides en utilisant une liste tabou [GL97] afin de limiter les risques de cycles durant la recherche. La fonction de voisinage est définie par l'ensemble des projections ayant une variable de plus ou de moins que la projection actuelle. Le passage d'un voisin à un autre se fait à l'aide de deux opérateurs : *ajout_var* qui ajoute aléatoirement une variable à la projection et *supprime_var* qui supprime la première variable permettant d'atteindre une projection valide. La recherche commence avec la projection de dimension maximale (toutes les variables) et applique *supprime_var* à chaque fois que cela est possible. La liste tabou permet d'éviter d'ajouter et de supprimer une variable plusieurs fois de suite.

Afin de permettre une exploration plus large de l'espace de recherche, plusieurs redémarrages de la recherche sont effectués. Pour ne pas reparcourir les mêmes projections, un mécanisme d'apprentissage permet de garantir un début de recherche différent pour chaque relance. Son principe est de donner un poids inversement proportionnel à l'ordre de sélection lors des recherches précédentes, pour chacune des variables, afin de pouvoir initier les relances suivantes en sélectionnant des variables ayant un poids faible. Ainsi, chaque relance oriente la recherche vers des zones de l'espace de recherche non explorées.

1.5 Résultats expérimentaux

Afin d'avoir un moyen de comparaison pour nos contributions, nous reproduisons ici les résultats fourni dans [CLSZ13] [4]. Les expérimentations sont faites sur une machine composé d'un processeur Intel Core^{tm} i7-2620M CPU à 2.70GHz (deux cœurs) avec 4 Go Ram tournant sous Linux 64-bits. L'option de compilation -Ofast est activé pour obtenir notre exécutable.

- Les instances réelles (raphv, raphy, rarep, rch8 et rch10) sont tirées de problèmes de caractérisation provenant de l'API BioMérieux basée sur des propriétés bio-chimiques de l'espèce Ralstonia et sur des expressions de gènes de virulences de l'espèce Xanthomonas.
- Les instances aléatoires sont générées en deux étapes, pour un nombre de variables et de groupes fixés. Tout d'abord, chaque entité est construite de manière aléatoire. Ensuite, une certaine proportion des entités est répartie de manière équitable entre tous les groupes et les entités restantes sont affectées une à une aléatoirement aux groupes. La proportion d'entités affectées aléatoirement est donnée en pourcentage et est appelée *bruit*. Une instance aléatoire est notée sous la forme : s*graine-bruit*.

Le temps autorisé pour chaque exécution est 10 minutes. Les résultats en gras indiquent que les solutions sont optimales.

Instances	Entités(SR)	Groupes	Gènes(SR)	PL	EPC(DIFF)(SH)	LSPC(DIFF)
s301-0	500	30	400	-	13	14
s326-0	500	10	500	-	13	14
s413-30	500	20	600	-	13	13 (14)
s555-20	800	20	800	-	13	13 (14)
s625-20	500	5	1000	-	13	13 (14)
s754-10	600	10	200	-	13	14
s882-20	600	10	400	-	13	14
s2501-70	800	10	800	-	15 (14)	15
s31294-50	200	15	1000	10	10	11
s3836-0	1000	15	1000	-	16	16
raphv	109 (108)	8	155 (68)	**6**	**6**	9
raphy	113 (112)	4	155 (70)	**6**	**6**	8
rarep	112	7	155 (72)	**12**	95 (59) (39)	14
rch8	132 (56)	21	37 (27)	**9**	**9**	9
rch10	173 (112)	27	98 (86)	**10**	27 (15) (25)	15

Colonne PL : Résultats obtenu par reformulation en programmation linéaire sur le solveur IBM *cplex* [5].Les instances marqués par "-" ne pas pu être chargé en mémoire car elles solicitaient plus de 32 Go de RAM.
SR : Obtenu après suppression des redondances.
DIFF : Résultats affichés dans l'article étant différents de ceux obtenus sur notre machine.
SH : Résultats obtenu sans heuristique avec notre programme sur notre machine.

4. Nous nous sommes servi du code mis à disposition sur http://forge.info.univ-angers.fr/~gh/Idas/Ccd/mcps/
5. http://www.ibm.com/software/integration/optimization/cplex-optimizer

Les différences significatives sur les instances rarep et rch10 sont du au fait que ces résultats ont été obtenu en partant d'une borne supérieur égale aux nombre de gènes divisé par deux. De fait , il apparaît clairement que l'instance rarep n'a pas bénéficié de la suppression de ses redondances [6].

Pour les comparaisons à venir sur les résultats qui diffèrent, nous prendrons en compte les résultats en bleu.

Pour l'instance rch10, le résultat sans heuristique étant meilleur que celui avec l'heuristique de EPC [7] sur notre machine, nous sommes en mesure de penser que le résultat indiqué dans l'article est erroné, donc nous choisissons de conserver le résultat sans heuristique.

6. $72/2 < 59$
7. Ce qui peut s'expliquer par une différence de structure et d'implémentation du code source.

2 Contributions

2.1 Introduction

Cette section présente les démarches de recherche qui ont été effectuées durant le stage.
Dans un premier temps, nous proposerons et définirons des critères qui permettent d'identifier si une instance est difficile ou non.
Ensuite, nous aborderons la résolution du MIN-PCM avec deux approches différentes :

— Une recherche exacte qui a la possibilité de prouver la borne minimum du MIN-PCM sur des instances de tailles raisonnables.
— Une recherche approchée qui à la possibilité de trouver des solutions de bonne qualité mais non nécessairement optimales, en un temps polynômial sur des instances de grandes tailles.

2.2 Définition d'une instance difficile

Prenons deux instances : une réelle (rch10) et une aléatoire (s3836-0), voici leurs caractéristiques :

Instances	Entités	Groupes	Gènes	Résolution PL	Résolution EPC [8]
s3836-0	1000	15	1000	-	16
rch10	173	27	98	**10** [9]	14

A priori, on peut supposer que l'instance aléatoire est plus difficile à résoudre : elle est bien plus volumineuse que l'instance réelle à tel point qu'elle nécessite plus de 32 Go de RAM pour une résolution en programmation linéaire.
Observons leurs résolutions avec notre algorithme sans heuristique présenté dans la figure 3 page 15

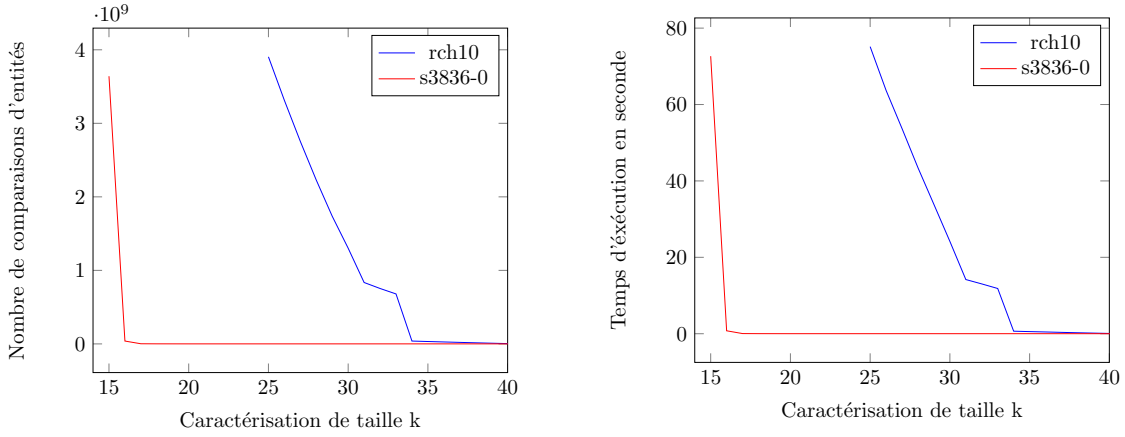

FIGURE 2 – Résolution sans heuristique de rch10 et s3836-0

Nous apercevons que l'instance aléatoire est facilement résolue jusqu'à une caractérisation de taille 15. Ce n'est pas le cas de l'instance réelle qui ne peut plus caractériser en un temps raisonnable à partir d'une caractérisation de taille 25. Ce type d'observation étant **systématique** quelque soit les caractéristiques des instances réelles ou aléatoires comparées, nous pouvons alors affirmer que la taille d'une instance ne suffit pas à elle seule pour définir sa difficulté. Dès lors, nous nous posons les deux questions suivantes :

— **Qu'est ce qui peut bien être à l'origine de cette différence de résolution entre une instance aléatoire et une instance réelle ?**
— **Existe il une méthode permettant de définir si une instance est difficile à résoudre ou non ?**

Afin de répondre à ces questions, nous définissons les notions suivantes :

Définition 11 *Le* ***masque*** *M* ***d'un groupe*** *g correspond à la moyenne des présences/absences des gènes pour chaque entité du groupe.*

Formellement, soit M_g le masque d'un groupe g, $g \in \mathcal{G}$, $M_g[i]$ la valeur du masque g en position i, $i \in [1, |\mathcal{X}|]$,

$$\forall i \in [1, |\mathcal{X}|], M_g[i] = \frac{\sum_{i=1}^{|\mathcal{G}|} e_i}{|\mathcal{G}|}$$

Définition 12 *Le **ratio** r **d'un masque** M correspond au pourcentage de valeur entière (0/1) présente dans le masque.*
Formellement, soit M_g le masque d'un groupe g, $r_g(I)$ le ratio du groupe g dans l'image I, $g \in \mathcal{G}$,

$$r_g(I) = \frac{|i/M_g[i] \in \{0,1\}|}{|\mathcal{X}|}, \forall i \in [1, |\mathcal{X}|]$$

Exemple 1 *Masque et ratio d'un groupe*

Gènes / *Entités*	*g0*	*g1*	*g2*	*g3*	*g4*	*g5*	*g6*	*g7*	*g8*	*g9*
e1	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*0*	*0*	*1*
e2	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*0*	*1*
e3	*1*	*1*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*
e4	*1*	*1*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*0*
e5	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*0*	*0*
Masque	*1*	*1*	*0*	*0.8*	*0.6*	*1*	*0*	*0.4*	*0*	*0.4*

Le ratio r de ce groupe est :
$r = 6/10$
soit $r = 0.6$

Définition 13 *L'**image** I **d'une instance** est une matrice en deux dimensions de taille $|\mathcal{G}| * |\mathcal{X}|$ où chaque ligne correspond au masque de chacun des groupes de l'instance.*
Formellement, soit I_g la ligne g de la matrice I correspondant à l'image de l'instance $\mathcal{I}$, M_g le masque du groupe g de l'instance $\mathcal{I}$,

$$\forall g \in [1, |\mathcal{G}|], I_g = M_g$$

Définition 14 *Le **taux de similarité** $\mathcal{T}_j$ **d'un gène** j correspond à la moyenne des valeurs de la colonne j sur l'image I d'une instance $\mathcal{I}$.*
Formellement, soit I l'image d'une instance $\mathcal{I}$, i la $i^{ème}$ ligne de I, j la $j^{ème}$ colonne de I, I_{ij} est la valeur dans I en ligne i et en colonne j, $\mathcal{T}_j(I)$ le taux de similarité du gène j dans l'image I,

$$Soit\ X = \frac{\sum_{i=1}^{|\mathcal{G}|} I_{ij}}{\mathcal{G}}$$

$$Si\ X < 0.5\ alors\ \mathcal{T}_j(I) = (0.5 - X) * 2$$

$$sinon\ \mathcal{T}_j(I) = (0.5 - (1 - X)) * 2$$

Ainsi formulé, $\mathcal{T}_j(I) \in [0, 1]$, et, plus le taux de similarité d'un gène est élevé, plus sa présence(resp. abscence) dans l'instance est redondante.

Définition 15 *Le **coefficient de difficulté** ρ **d'une instance**, correspond à la moyenne des taux de similarité $\mathcal{T}$ d'une instance.*
Formellement, soit I l'image d'une instance $\mathcal{I}$, j la $j^{ème}$ colonne de I, $\mathcal{T}_j(I)$ le taux de similarité globale du gène j dans l'image I,

$$\rho = \frac{\sum_{j=1}^{|\mathcal{X}|} \tau_j(I)}{|\mathcal{X}|}$$

Définition 16 *Le **coefficient de difficulté** σ **d'une instance**, correspond au complémentaire de la moyenne des ratios des masques d'une instance*[10]*. Nous utilisons le complémentaire de façon à ce que l'interprétation du coefficient σ soit calqué sur celui de ρ. Formellement, soit I l'image d'une instance $\mathcal{I}$, i la $i^{ème}$ ligne de I, $r_i(I)$ le ratio du groupe i dans l'image I,*

$$\sigma = 1 - \frac{\sum_{i=1}^{|\mathcal{G}|} r_i(I)}{|\mathcal{G}|}$$

Définition 17 ***Le coefficient $\Delta\mathcal{T}$ d'une instance*** *correspond à l'écart type des taux de similarité $\mathcal{T}$ des gènes. Formellement, soit I l'image d'une instance $\mathcal{I}$, i la $i^{ème}$ ligne de I, $r_i(I)$ le ratio du groupe i dans l'image I, le coefficient de difficulté ρ de l'instance $\mathcal{I}$,*

$$\Delta\mathcal{T} = \sqrt{\frac{\sum_{j=1}^{|\mathcal{X}|}(\rho - \mathcal{T}_j)^2}{|\mathcal{X}|}}$$

$$\Delta\mathcal{T} \in [0, \frac{1}{2}]$$

Remarque 1 *Si $\rho \simeq 0$ (resp. 1) alors $\Delta\mathcal{T} \simeq 0$ car ρ est la moyenne des taux de similarité $\mathcal{T}$ d'une instance et si cette moyenne est proche de 0 (resp. 1), cela signifie que la majorité des $\mathcal{T}$ sont proches de 0 (resp. 1) et donc que leur écart type ($\Delta\mathcal{T}$) est proche de 0.*

Reprenons nos deux instances rch10 et s3836-0 et calculons leurs coefficients de difficultés :

Instances	$\Delta\mathcal{T}$	ρ	σ
s3836-0	0.019	0	0.024
rch10	0.238	0.626	0.906

Nous observons que le coefficient de difficulté ρ semble plus significatif que σ pour déterminer la difficulté d'une instance, mais nous ne sommes pas en mesure d'indiquer dans quel proportion. Cependant les travaux de [CLSZ13] nous indiquent que seule une heuristique sur le choix des variables est en mesure de pouvoir améliorer un algorithme de recherche exacte. Cela nous conforte dans l'idée que ρ a plus d'influence que σ sur la difficulté d'une instance. Nous pouvons ainsi formuler les deux propositions suivantes :

Proposition 2 *Une instance dont le coefficient de difficulté ρ est proche de 1 est une **instance difficile** à résoudre.*

Proposition 3 *Une instance dont le coefficient de difficulté ρ est proche de 1 et dont le coefficient de difficulté σ est proche de 1 est une **instance très difficile** à résoudre.*

Dès lors, nous pourrions penser qu'il suffit de trouver une heuristique basée sur le choix des gènes en fonction de leurs taux de similarité $\mathcal{T}$ pour outrepasser la difficulté émise par le coefficient ρ. Ce n'est pas le cas. En effet, si l'écart type entre les différents taux $\mathcal{T}$ d'une instance est faible, alors le critere $\mathcal{T}$ ne permet pas de différencier significativement les gènes. Nous faisons donc la proposition suivante :

Proposition 4 *Une heuristique basée sur le choix des gènes en fonction de leurs taux de similarité $\mathcal{T}$ n'a aucune influence sur des instances dont le coefficient $\Delta\mathcal{T}$ est proche de 0.*

Il y a encore au moins un indicateur sur la difficulté des instances : il s'agit de la taille de celles-ci. Nous rejoignons l'idée émise dans [CLSZ13], selon laquelle la taille d'une instance est caractérisée par son nombre de gènes et d'entités mais pas par son nombre de groupe. Cependant, lorsque le coefficient de difficulté σ est proche de 0, nous devrions pouvoir être en mesure d'outrepasser cette difficulté puisque nous pouvons réduire le parcours d'un très grand nombre d'entités par l'utilisation des masques, nous présentons ce cas de figure dans l'exemple 2. Nous faisons donc la proposition suivante :

Proposition 5 *La taille d'une instance est une information sur la difficulté de sa résolution. Il n'existe pas d'heuristique permettant d'outrepasser cette difficulté lorsque le coefficient σ de l'instance est proche de 1.*

10. La moyenne des ratios des masques d'une instance $\in [0, 1]$

Exemple 2 (Difficultés d'une instance) *Supposons qu'il existe une instance composé de 3 groupes de 100 entités que l'on souhaite caractériser avec un ensemble de n gènes. Supposons également que cette instance ait pour coefficients de difficulté* $\sigma \simeq 0$, $\rho \simeq 0.7$ *et* $\Delta\mathcal{T} \simeq 0$. *D'après notre proposition 4, il n'existe pas d'heuristique efficace pour outrepasser la difficulté émise par* ρ.

Une recherche standard pour une caractérisation de taille k effectue au maximum environ $(100 * 200 + 100 * 100) * C_n^k * k$ *comparaisons. Comme sigma* $\simeq 0$ *, nous pouvons exploiter les masques(voir paragraphe 2.3.5 page 21). En les utilisant, nous effectuons au maximum environ* $(1^+ * 2^+ + 1^+ * 1^+) * C_n^k * k$ *comparaisons.*

On remarque que dans ce cas précis, l'utilisation des masques à une influence sur la résolution du problème.

Supposons maintenant que $\Delta\mathcal{T} \simeq 0.5$, *nous présumons dès lors que l'efficacité d'une heuristique basée sur l'utilisation des masques serait bien dérisoire face à une heuristique basée sur le choix des gènes en fonction de leurs taux de similarité* $\mathcal{T}$ [11]. *En effet, une heuristique basée sur le choix des gènes en fonction de leurs taux de similarité* $\mathcal{T}$ *travaille à réduire le facteur à caractère exponentiel* C_n^k *alors que l'utilisation des masques vise à réduire un facteur à caractère polynômial. Cela confirme une fois de plus que* ρ *à plus d'influence que* σ *pour caractériser une instance.*

2.2.1 Observations

Nous présentons ici les instances avec leurs coefficients de difficultés respectifs :

Instances	Entités	Gènes	$\Delta\mathcal{T}$	ρ	σ	PL	EPC	LSPC
s301-0	500	400	0.025	0.034	0.999	-	13	14
s326-0	500	500	0.026	0.033	1	-	13	14
s413-30	500	600	0.027	0.035	1	-	13	13
s555-20	800	800	0.029	0.039	0.999	-	13	13
s625-20	500	1000	0.027	0.035	1	-	13	13
s754-10	600	200	0.027	0.034	1	-	13	14
s882-20	600	400	0.024	0.032	1	-	13	14
s2501-70	800	800	0.024	0.033	1	-	15	15
s31294-50	200	1000	0.049	0.065	0.993	10	10	11
s3836-0	1000	1000	0.019	0.024	1	-	16	16
raphv	108	68	0.302	0.588	0.419	**6**	**6**	9
raphy	112	70	0.294	0.609	0.668	**6**	**6**	8
rarep	112	72	0.295	0.651	0.502	**12**	39	14
rch8	56	27	0.339	0.569	0.067	**9**	**9**	9
rch10	112	86	0.238	0.626	0.094	**10**	25	15

Toute les instances aléatoires ont un coefficient ρ proche de 0 et donc un coefficient $\Delta\mathcal{T}$ proche de 0 (voir remarque 1). Mais elles ont pour difficulté leurs coefficients σ qui est proche de 1, cela signifie que nous ne pourrons pas réduire de façon significative leurs temps de résolution. L'instance rch8 a un coefficient σ proche de 0 ainsi qu'un coefficient ρ moyennement élevé, de plus, elle est de faible taille, c'est l'instance qui semble la plus facile à résoudre. L'instance raphv et raphy ont un ordre de difficulté similaire, L'instance rch10 a un très faible coefficient σ, elle doit être plus facile à résoudre que l'instance rarep pour qui le coefficient sigma est moyennement élevé. Ces deux dernières semblent être les instances les plus difficiles à résoudre. Cette série d'observations et de raisonnement est corroborée par les résultats obtenus par [CLSZ13].

2.3 Recherche exacte

2.3.1 Introduction

Nous présentons les heuristiques ayant été mises en place pour la résolution d'instance MIN-PCM. Chaque heuristique est abordée de façon indépendante. Une comparaison entre toutes les heuristiques est présentée dans la sous section 2.3.6. Les comparaisons se font sur l'instance réelle rch10 et l'instance pseudo-aléatoire s3836-0.

Une comparaison entre la meilleure combinaison d'heuristique obtenue et l'heuristique "CCD" proposée par [CLSZ13] avec le solveur *Exact-Proj-Car* est présentée à la fin de la cette section.

11. Quoique cette assertion dépend de la valeur de n

2.3.2 Algorithmes généraux de résolution

Dans cette sous section, nous présentons une méthode générale qui permet de résoudre le MIN-PCM.

```
booléen minimise_caractérisation_instance (Groupes, n)
// Groupes est un ensemble contenant tous les groupes de l'instance
// n correspond aux nombres de gènes de l'instance
caracterise ← VRAI
k ← n // k correspond au nombre de gènes pouvant caractériser l'instance
tant que caracterise = VRAI faire
    k ← k − 1
    caracterise ←caractérise_instance (Groupes, k, n)
fin
afficher("La caractérisation minimale est de taille " k + 1)
```

FIGURE 3 – Algorithme de minimisation du problème de caractérisation multiple

```
booléen caractérise_instance (Groupes, k, n)
// Groupes est un ensemble contenant tous les groupes de l'instance
// k correspond au nombre de gène souhaité pour la caractérisation
// n correspond aux nombres de gènes de l'instance
pour tout combinaison de C_n^k faire
    DejaVus ← {}
    caracteriseTemp ← VRAI
    pour tout gRef ∈ Groupes faire
        DejaVus ← DejaVus ∪ {gRef}
        pour tout gComp ∈ Groupes\DejaVus faire
            si ¬ caractérise_groupes(combinaison,gRef,gComp) alors
                caracteriseTemp ← FAUX
                Sortir de la boucle
            fin
        fin pour
        si caracteriseTemp =FAUX alors
            Sortir de la boucle
        fin
    fin pour
    si caracteriseTemp = VRAI alors
        retourner VRAI
    fin
    // Sinon combinaison suivante
fin pour
afficher("Il n'existe pas de caractérisation de taille " k)
retourner FAUX
```

FIGURE 4 – Algorithme de caractérisation d'une instance MIN-PCM pour une taille k

L'algorithme présenté dans la figure 3 cherche la plus petite caractérisation possible pour une instance donnée. Il prend en entrée un ensemble de groupe *Groupes* qui contient tous les groupes de l'instance et un entier n qui correspond aux nombres de gènes de l'instance. Au début de l'algorithme, une variable booléenne *caracterise* est instanciée à $VRAI$ et un entier k, qui correspondra au nombre de gènes pouvant caractériser une instance, est instancié à n. En effet, la caractérisation la plus évidente et immédiate est la caractérisation de taille n. Nous entrons ensuite dans une boucle qui va avoir pour rôle de minimiser le plus possible la variable k. On sort de cette boucle lorsque plus aucune caractérisation n'est possible, c'est à dire lorsque nous avons trouvé la borne minimale du MIN-PCM. Pour cela, il faut que la variable *caracterise* soit instanciée à faux, ceci ne peut se faire que par l'appel à l'algorithme *caractérise_instance* présenté dans la figure 4.

Cet algorithme prend en entrée un ensemble de groupe *Groupes* qui contient tous les groupes de l'instance, un entier k, qui correspond au nombre de gènes souhaités pour la caractérisation et un entier

```
booléen caractérise_groupes (combinaison,g1,g2)
// e1 et e2 sont les entités des groupes g1 et g2
pour tout e1 ∈ g1 faire
    pour tout e2 ∈ g2 faire
        temp ← FAUX
        pour tout i ∈ combinaison faire
            si e1[i] ≠ e2[i] alors
                temp ← VRAI
                Sortir de la boucle
            fin
        fin pour
        si temp =FAUX // g1 et g2 ne sont pas caractérisable avec combinaison
        alors
            retourner FAUX
        fin
    fin pour
fin pour
retourner VRAI
```

FIGURE 5 – Algorithme de caractérisation de groupes

n qui correspond au nombre de gènes de l'instance. Nous entrons dans une première boucle (niveau 1) qui génère les combinaisons *combinaison* de C_n^k. La variable $DejaVus$ est instanciée comme étant un ensemble vide de groupe. Une variable booléenne $caracteriseTemp$ est initialisée à $VRAI$. Nous entrons alors dans une seconde boucle (niveau 2), qui va ouvrir une nouvelle boucle (niveau 3) afin de tester si tous les groupes de *Groupes* peuvent caractériser les groupes de $Groupes \backslash DejaVus$, auquel cas, nous avons une caractérisation de taille k avec la combinaison *combinaison*. La mise à jour de $DejaVus$ au niveau 2 a pour effet de ne pas tester plusieurs fois des groupes ayant déjà été testés. Dès lors que la variable $caracteriseTemp$ est instanciée à $FAUX$, un mécanisme de sortie de boucle permet de tester directement la prochaine combinaison de C_n^k. Si en sortie de boucle de niveau 2, la variable $caracteriseTemp$ est toujours instanciée à $VRAI$, cela signifie que la combinaison courante caractérise l'instance, l'algorithme s'arrête alors en retournant la valeur booléenne $VRAI$. Si nous arrivons en sortie de boucle de niveau 1, cela signifie que toutes les combinaisons de C_n^k ont été testées et que aucune d'entre elles n'a pu caractériser l'instance. Dans ce cas, l'algorithme s'arrête en retournant la valeur booléenne $FAUX$. Dans cet algorithme, nous faisons appel à la fonction *caractérise_groupes* pour savoir si deux groupes différents peuvent être caractérisés avec une combinaison *combinaison*. Nous présentons cette fonction dans la figure 5.

Cette fonction permet de savoir si deux groupes $g1$ et $g2$ sont caractérisables avec une combinaison *combinaison* donnée. Nous notons $e1$(resp.$e2$) la variable qui va contenir une après l'autre les entités du groupe $g1$(resp.$g2$). Les entités de chacun des groupes ($e1$ et $e2$) sont comparées deux à deux sur la présence/absence des gènes définit par la combinaison *combinaison*. Si deux entités sont identiques sur ces gènes alors on ne peut pas caractériser les deux groupes avec cette combinaison(et a fortiori, on ne peut pas non plus caractériser l'instance avec cette combinaison). Dès lors, l'algorithme s'arrête immédiatement en retournant la valeur boolénne $FAUX$. Si toutes les comparaisons possibles entre $e1$ et $e2$ ont été faites, cela signifie que les deux groupes $g1$ et $g2$ sont caractérisables avec la combinaison *combinaison*. L'algorithme s'arrête alors en retournant la valeur booléenne $VRAI$.

2.3.3 Heuristique de tri sur les gènes

Tri des gènes par taux de similarité $\mathcal{T}$ Les outils mis en place dans la sous-section 2.2 nous permettent de mettre en place une heuristique basée sur le trie des gènes : nous ordonnons les gènes par ordre croissant sur leur taux de similarité $\mathcal{T}$. Dès lors, nous parcourons en priorité les gènes présentants un faible taux de similarité. Ce trie par $\mathcal{T}$ permet d'explorer en priorité les gènes susceptible de caractériser une instance. Le coût de calcul de ce trie est insignifiant et est effectué de façon unique avant le lancement de l'algorithme de résolution.

Résultats Sur l'instance réelle rch10, nous constatons que le nombre de comparaisons d'entités et le temps d'éxécution **sont considérablement diminuées** par l'heuristique(figure 6). Sur l'instance

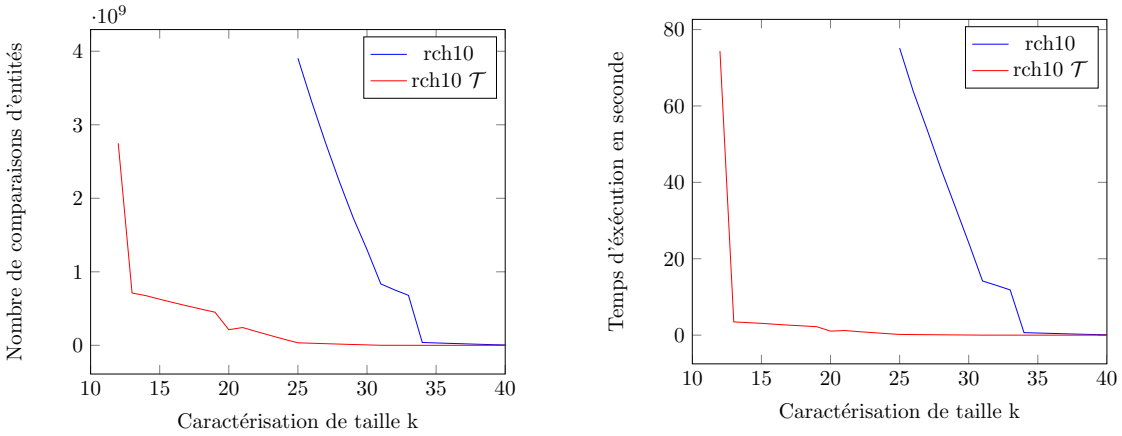

FIGURE 6 – Heuristique $\mathcal{T}$ sur rch10

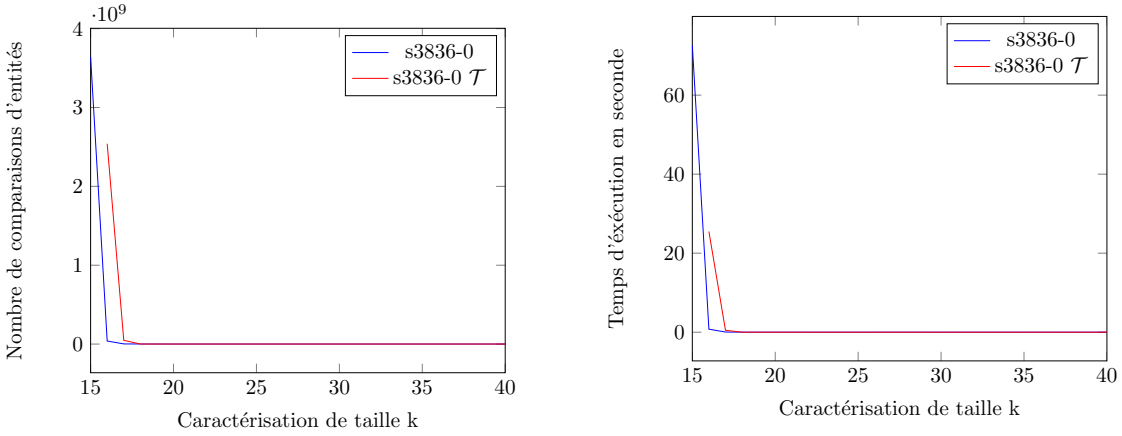

FIGURE 7 – Heuristique $\mathcal{T}$ sur s3836-0

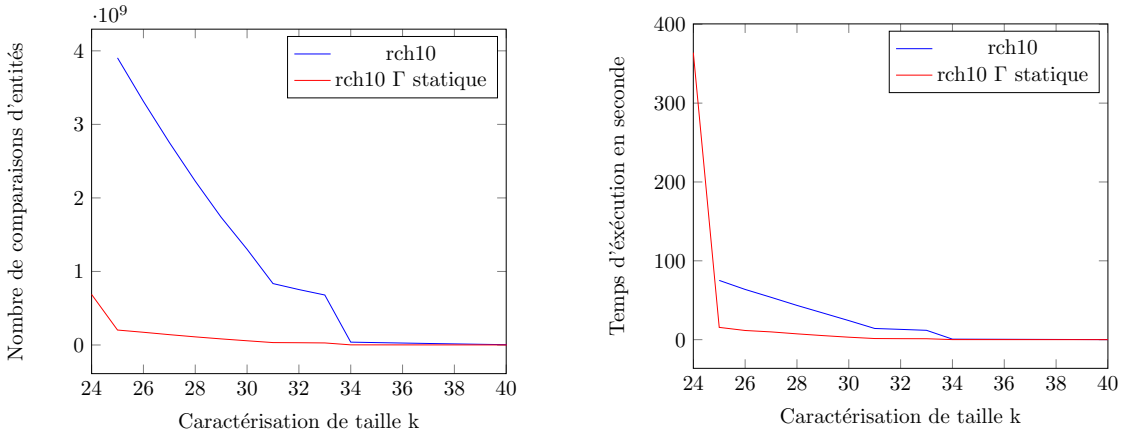

FIGURE 8 – Heuristique Γ statique sur rch10

aléatoire, nous obervons que cette heuristique a une faible influence négative(figure 7). Ce phénomène s'explique par un coefficient $\Delta\mathcal{T}$ très bas (0.019). Il résulte de cette observation que **toute instance ayant un coefficient $\Delta\mathcal{T} \simeq 0$ ne pourra pas bénéficier, lors de sa résolution, des avantages d'une heuristique de tri sur les gènes**.

La possibilité que l'influence soit négative nous permet de déduire que **le tri des gènes par $\mathcal{T}$ n'est pas un tri optimal**.

2.3.4 Heuristique de tri sur les groupes

Tri des groupes par taux de similarité Γ Lors de la précédente sous-section, nous avons ordonnés statiquement les gènes d'après leurs $\mathcal{T}$. Ici, nous souhaitons ordonner les entités d'une instance. Nous définissons alors le concept de taux de similarité entre deux groupes.

Définition 18 *Le **taux de similarite** $\Gamma(g, g')$ **entre deux groupes** g **et** g' correspond à la moyenne des sommes des moyennes des valeurs du masque de g et g'.*
Formellement, soit g et g' deux groupes d'une instance $\mathcal{I}$, M_g(resp. $M_{g'}$) le masque du groupe g(resp. g'), $M_g[i]$(resp. $M_{g'}[i]$) la valeur du masque du groupe g(resp. g') en position i,

$$\Gamma(g, g') = \frac{\sum_{i=1}^{|\mathcal{X}|}(\frac{M_g[i]+M_{g'}[i]}{2})}{|\mathcal{X}|}$$

Une instance est caractérisable si tous les groupes qui la composent sont caractérisables entre eux. Nous proposons de regarder en priorité les groupes ayant le plus de chance de ne pas caractériser l'instance car dès lors que deux groupes ne sont pas caractérisables entre eux, nous pouvons arrêter la recherche sur la combinaison courante et tester la suivante, nous faisons ainsi l'économie de tester les groupes facilement caractérisables entre eux(qui sont de toute manière testés lorsque la combinaison courante est une caractérisation valide). Nous construisons alors une liste de paire de groupe $[g, g']$ de tel manière que la liste soit triée par ordre décroissant de $\Gamma(g, g')$ et qu'elle contienne tous les groupes qu'il faut caractériser entre eux pour caractériser une instance. En effet, plus deux groupes sont similaires entre eux, plus leur chance de ne pas caractériser l'instance est forte.

Nous proposons cette heuristique de façon statique (tout les gènes sont pris en compte, la liste est créée de façon unique avant le lancement de l'algorithme) et de façon dynamique(à chaque combinaison, seuls les gènes concernés par la combinaison sont pris en compte pour le calcul de $\Gamma(g, g')$, nous construisons donc autant de liste qu'il y a de combinaisons testées)

Résultats Sur la figure 8, nous observons que l'heuristique Γ statique permet de réduire considérablement le nombre de comparaisons d'entité. Le temps de résolution est également amélioré. Lorsque nous traitons cette instance avec l'heuristique Γ dynamique(figure 9), nous apercevons que le nombre de comparaisons d'entité est également plus faible, mais nous payons le caractère dynamique de l'heuristique puisque le temps de résolution est beaucoup plus élevé que la résolution standard. La borne inférieure est par voie de conséquence très élevée ($k = 32$ contre $k = 25$ en résolution standard). Nous nous intéressons dès lors à comparer les deux heuristiques sur l'instance. nous constatons que Γ dynamique est meilleure

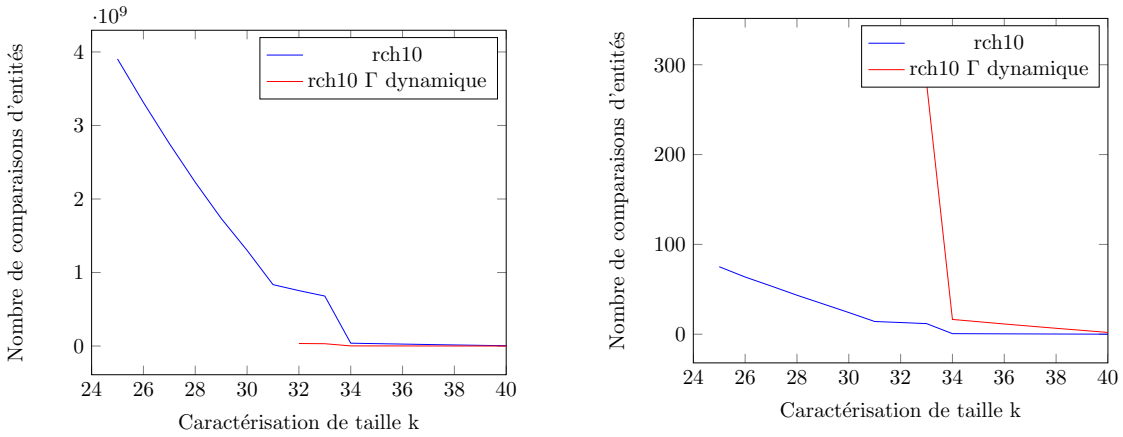

FIGURE 9 – Heuristique Γ dynamique sur rch10

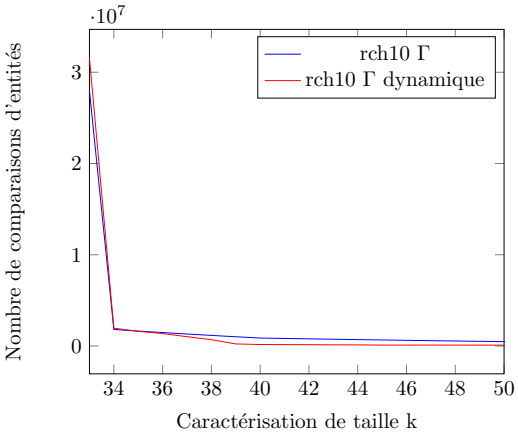

FIGURE 10 – Zoom sur comparaisons heuristique Γ statique et heuristique Γ dynamique

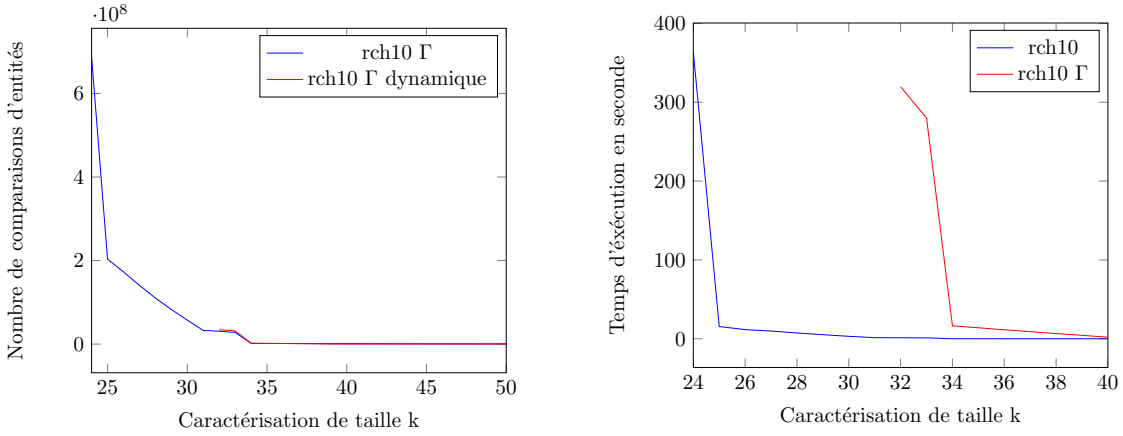

FIGURE 11 – Comparaisons heuristique Γ statique et heuristique Γ dynamique sur rch10

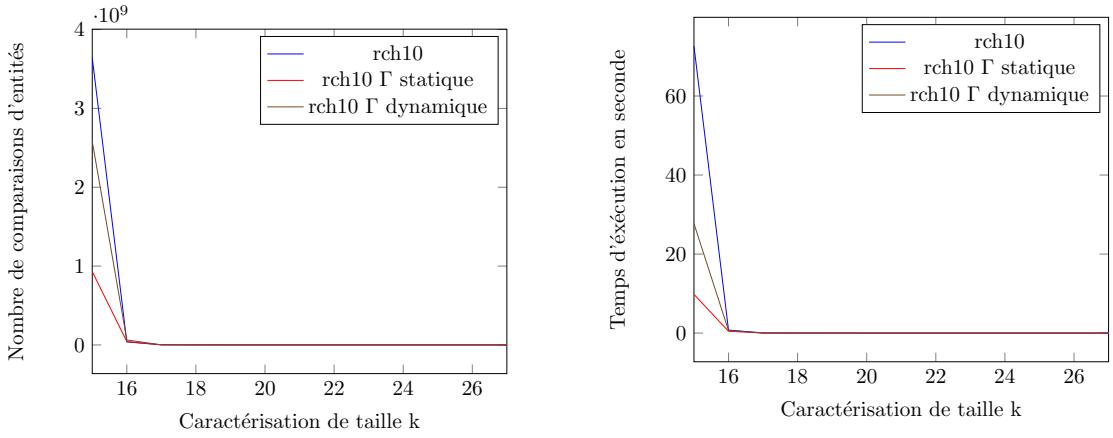

FIGURE 12 – Comparaisons heuristique Γ statique et heuristique Γ dynamique sur s3836-0

que Γ statique sur le nombre de comparaisons d'entité jusque $k = 35$ (figure 10), mais que passé cette borne, Γ statique est bien plus performante(figure 11) aussi bien en terme de nombre de comparaison d'entité que en terme de temps de résolution. Les observations sur l'instance aléatoire n'étant pas beaucoup plus fructueuse pour l'heuristique Γ dynamique (figure 12), nous concluons que cette dernière n'est pas à retenir pour les résolutions à venir, au contraire de l'heuristique Γ statique qui, pour un coût de calcul insignifiant, nous réduit le temps de résolution de manière conséquente.

La figure 10 nous prouve également que aucune des deux heuristiques n'effectue un tri optimal.

2.3.5 Autres heuristiques

Heuristique des plus mauvais d'abord (pmda) Lorsque nous parcourons une instance pour une caractérisation de taille k, nous affectons un poids sur la paire d'entités comparées deux à deux sur les indices de la combinaison courante.

Définition 19 (Poids P d'une paire d'entités) *Formellement, soit e_a, e_z deux entités n'appartenant pas au même groupe, $\mathcal{C}$ une combinaison de $\mathcal{C}^k_{|\mathcal{X}|}$, P le poids de la paire $\{e_a, e_z\}$.*

$$P = |\{i / e_a(i) = e_z(i), \forall i \in \mathcal{C}\}|$$

Si $P >= k - 1$ alors la ***paire d'entités*** *$\{e_a, e_z\}$* ***est critique*** *car lors du parcours de la prochaine combinaison, cette paire a la plus forte probabilité d'être identique sur les nouveaux indices.*

L'idée est donc de traiter au plus vite ce type d'entités afin d'examiner au plus vite d'autres combinaisons.

Exemple 3 *Soit les 2 entités suivantes :*

Groupe	*Gènes* / *Entités*	*g1*	*g2*	*g3*	*g4*	*g5*	*g6*
1	*e1*	*1*	*0*	*1*	*0*	*0*	*0*
20	*e400*	*1*	*0*	*0*	*0*	*0*	*1*

Supposons que nous sommes dans le cas d'une caractérisation de taille 3, nous parcourons les combinaisons de $\mathcal{C}^3_6$. Supposons que le groupe 1 soit de taille 1, si nous sommes à la comparaison entre e1 et e400, cela signifie que nous avons déjà effectué 399 comparaisons d'entités. Observons la résolution de cet exemple :

- *Regardons alors la combinaison courante $\mathcal{C} = \{123\}$: nous apercevons que seul g3 permet la caractérisation. Comme plusieurs des combinaisons suivantes ne différeront que d'un élément, cet paire d'entités {e1,e400} a la plus forte probabilité d'être similiraire lors de la prochaine combinaison, nous gardons donc en mémoire cet ensemble qui a un poids égale à $k - 1$.*
- *Supposons que $\mathcal{C} = \{123\}$ n'ai pas caractérisé notre instance, nous parcourons alors $\mathcal{C} = \{124\}$: nous commençons par parcourir les ensembles critiques obtenus lors du parcours précédent, soit la comparaison entre e1 et e400. Le poids est alors égal à k, ce qui signifie que nous pouvons arrêter notre recherche sur cette combinaison (car celle ci ne pourra en aucun cas caractériser l'instance). Cependant nous gardons en mémoire cet ensemble critique. Notons que nous avons fait là l'économie de 399 comparaisons d'entités.*

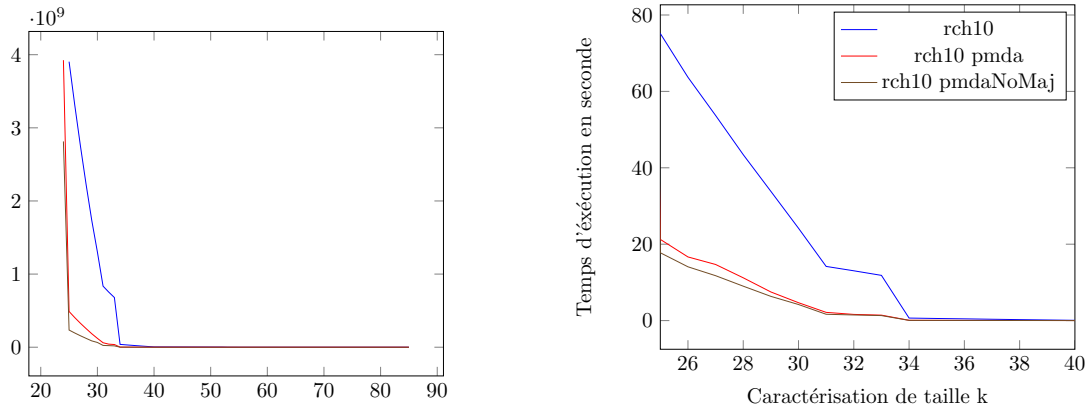

FIGURE 13 – Heuristiques pmda sur rch10

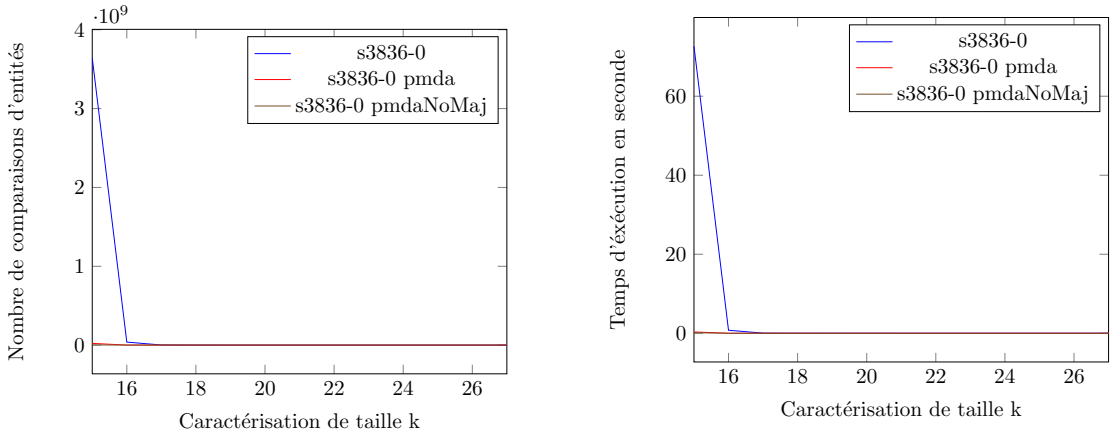

FIGURE 14 – Heuristiques pmda sur s3836-0

— *Nous parcourons alors* $\mathcal{C} = \{125\}$ *: même constat , de nouveau une économie de 399 comparaisons d'entités.*
— *Nous parcourons alors* $\mathcal{C} = \{126\}$ *: aucun effet, mais la paire* $\{e1,e400\}$ *est toujours considérée comme critique.*
— *Nous parcourons alors* $\mathcal{C} = \{134\}$ *: aucun effet, mais la paire* $\{e1,e400\}$ *n'est plus considérée comme critique car son poids* $< k - 1$.

Nous proposons une variante à cette heuristique pour laquelle dans le dernier point de l'exemple 3, nous continuons de considérer la paire d'entité comme critique(i.e. : une paire d'entité qui a été définit comme étant critique ne peut plus changer de statut). Nous appelons cette variante pmdaNoMaj [12].

Résultats Nous constatons que les deux heuristiques sont efficaces sur les deux instances(figure 13 et 14). L'heuristique pmdaNoMaj est légèrement plus efficace que pmda sur l'instance rch10(cela se confirme sur d'autres instances).

Heuristique des valeurs taboues Nous avons vu dans l'exemple 2 page 14 qu'il est possible d'utiliser les propriétés des masques pour améliorer le temps de résolution d'une instance MIN-PCM. L'heuristique des valeurs taboues permet justement de tirer partie de ces propriétés. Cette heuristique ne pourra fonctionner que sur des instances dont le coefficient $\sigma \simeq 0$.

Exemple 4 *Heuristique des valeurs taboues Considérons les masques des deux groupes suivants :*

12. Plus mauvais d'abord sans mise à jour

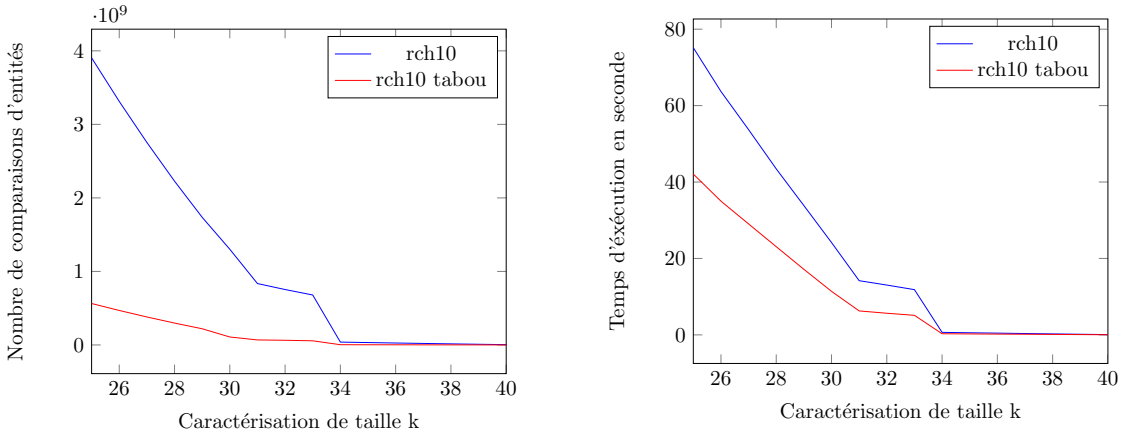

FIGURE 15 – Heuristique des valeurs taboues sur rch10

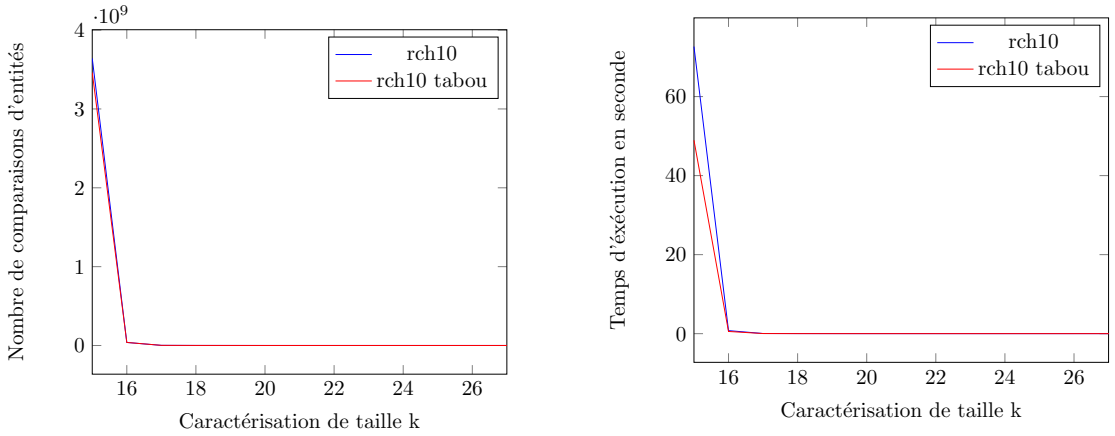

FIGURE 16 – Heuristique des valeurs taboues sur s3836-0

Gènes / *Groupe*	*g0*	*g1*	*g2*	*g3*	*g4*
Gr1	*1*	*0.8*	*1*	*0*	*1*
Gr2	*1*	*1*	*0.7*	*0*	*0*

Lorsque nous essayons de caractériser le groupe Gr1 et Gr2 avec une combinaison de gènes, nous sommes obligés de comparer chaque entité avec les gènes présent dans la combinaison dont les valeurs dans les masques de Gr1 et Gr2 ne correspondent pas à des valeurs entières(0 ou 1). Mais dans le cas où ces valeurs correspondent à des valeurs entières, nous pouvons considérer les deux masques comme des entités propres puisque toutes les entités des groupes présenteront la même valeur sur les gènes en question. Nous pouvons ainsi faire l'économie du parcours des combinaisons des différentes entités du groupe Gr1 et Gr2, ces valeurs n'ayant pas besoin d'être observées, nous considérons qu'elles sont taboues. Dans notre exemple, nous devrons parcourir de façon exhaustive les colonnes des gènes g1 et g2 lorsque celles-ci feront partie d'une combinaison à tester. Mais ce ne sera pas le cas pour les colonnes des autres gènes (g0, g3 et g4) pour lesquelles nous avons une comparaisons directe.

Résultats Nous observons sur la figure 15 que l'heuristique permet de réduire le nombre de comparaisons d'entités ainsi que le temps de résolution sur l'instance réelle dont le coefficient $\sigma = 0.093$. Ce n'est pas le cas pour l'instance aléatoire (figure 16) dont le coefficient $\sigma = 1$.

2.3.6 Comparaisons entre heuristique

Nous comparons ici les meilleures heuristiques retenues dans cette sous section afin d'identifier lesquelles sont les plus influentes lors de la résolution d'une instance MIN-PCM. Nous ajoutons également une heuristique qui combine les heuristiques retenues(all).

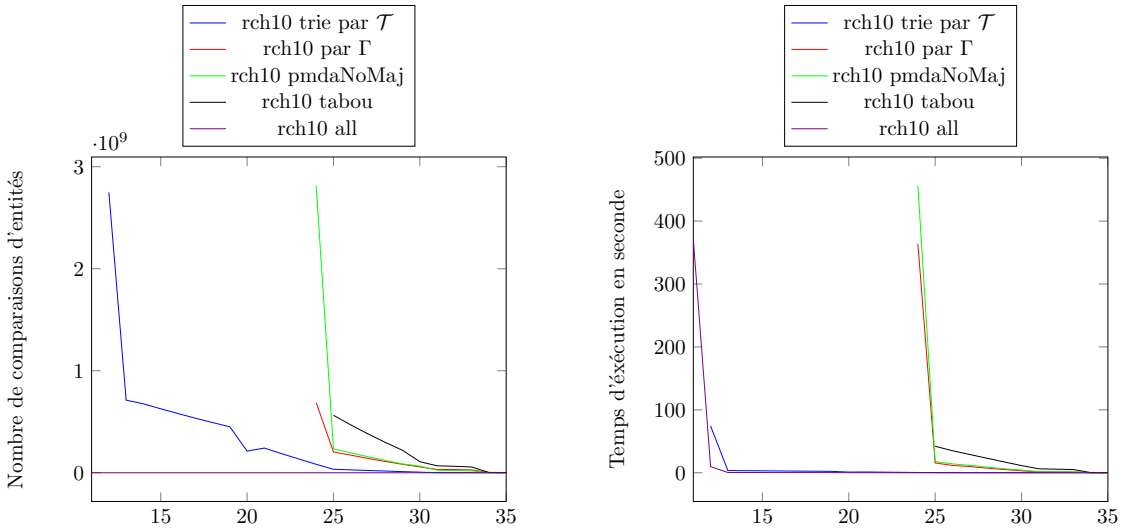

FIGURE 17 – Comparaisons des heuristiques sur rch10

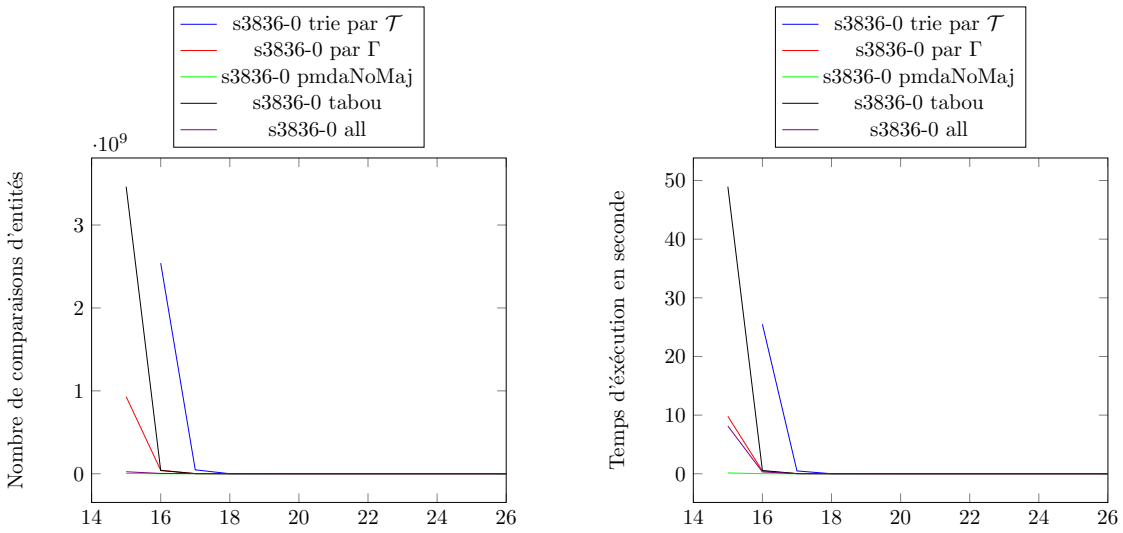

FIGURE 18 – Comparaisons des heuristiques sur s3836-0

Nous observons sur la figure 17 que l'heuristique de tri par taux de similarité $\mathcal{T}$ (en bleu) est la plus influente des heuristiques. Vient ensuite l'heuristique de tri des groupes par Γ (en rouge), celle-ci a une influence quasi similaire avec l'heuristique des plus mauvais d'abord sans mise à jour (pmdaNoMaj, en vert). L'heuristique des valeurs taboues(en noir) est la moins influente. Nous constatons que la combinaison de ces heuristiques (en violet) permet d'obtenir une excellente heuristique.

Sur la figure 18, l'heuristique de tri par taux de similarité $\mathcal{T}$ (en bleu) est la moins influente, cela est dû au coefficient $\Delta\mathcal{T} \simeq 0$. L'heuristique des plus mauvais d'abord sans mise à jour (pmdaNoMaj, en vert) surpasse toutes les heuristiques, y compris l'heuristique combinant toutes les autres (en violet), nous pensons qu'il s'agit là du coût de calcul de l'heuristique des valeurs taboues (en noir) et qui n'a pas beaucoup d'influence du fait que le coefficient de difficulté $\sigma = 1$. L'heuristique de tri des groupes par Γ (en rouge) apporte une amélioration significative.

Nous concluons que les heuristiques choisies pour la résolution d'une instance MIN-PCM doivent l'être en fonction des différents coefficients ($\Delta\mathcal{T}$, σ, ρ) de l'instance.

2.3.7 Résultats

Nous présentons ici les résultats obtenus avec notre solveur *Exact-Proj-Car2* (EPC2) qui fonctionne avec toutes les meilleures heuristiques présentées dans cette sous section. Nous pouvons ainsi comparer nos résultats avec ceux obtenus par [CLSZ13].

La colonne *temps* indique le temps cumulé depuis le début de la recherche à $k = |\mathcal{X}|$.

Instances	Entités	Gènes	$\Delta\mathcal{T}$	ρ	σ	PL	EPC	EPC2	
								k	temps
s301-0	500	400	0.025	0.034	0.999	-	13	13	2.072
s326-0	500	500	0.026	0.033	1	-	13	13	1.707
s413-30	500	600	0.027	0.035	1	-	13	13	1.050
s555-20	800	800	0.029	0.039	0.999	-	13	13	14.924
s625-20	500	1000	0.027	0.035	1	-	13	13	1.636
s754-10	600	200	0.027	0.034	1	-	13	14	0.366
s882-20	600	400	0.024	0.032	1	-	13	13	543.176
s2501-70	800	800	0.024	0.033	1	-	15	14	282.935
s31294-50	200	1000	0.049	0.065	0.993	10	10	10	381.108
s3836-0	1000	1000	0.019	0.024	1	-	16	15	13.072
rch8	56	27	0.339	0.569	0.067	**9**	**9**	**9**	0.038
raphv	108	68	0.301	0.588	0.419	**6**	**6**	**6**	2.055
raphy	112	70	0.293	0.609	0.668	**6**	**6**	**6**	6.846
rarep	112	72	0.295	0.651	0.502	**12**	39	16	206.037
rch10	112	86	0.237	0.626	0.094	**10**	25	11	383.006

Nous avons placé en bleu les caractérisations de taille inférieure à EPC. Les temps de couleur cyan permettent de souligner la rapidité d'éxécution de EPC2 par rapport à EPC : EPC2 caractérise l'instance s3836-0 en 13 secondes pour un k=15 alors que EPC ne réduit pas en dessous de k=16 en 10 minutes. Nous observons le même type de phénomène sur l'instance s2501-70. Notons toutefois que l'instance s754-10 est moins bien caractérisée par EPC2, nous pensons que dans ce cas précis, l'influence du tri par $\mathcal{T}$ est négative, cela est du à $\Delta\mathcal{T}$ proche de 0. Notre solveur réduit de façon significative les caractérisations des instances réelles rarep et rch10.

2.4 Recherche incomplète

Nous nous intéressons ici à obtenir une méthode de recherche incomplète qui se sert des propriétés révélées dans la sous section 2.2. Nous mettons un mécanisme de roulette proportionnelle (non adaptative) sur les gènes en fonction de leur taux de similarité $\mathcal{T}$: lors d'un tirage, chaque gène a une probabilité inversement proportionelle à son taux de similarité d'être sélectionné.

Définition 20 *La probabilité $P(i)$ du gène i d'être sélectionné lors d'un tirage se calcule de la façon suivante :*
Soit $i \in [1, \ldots, |\mathcal{X}|]$, les gènes i sont ordonnées par ordre croissant de similarité $\mathcal{T}$,

$$P(i) = \frac{1 - \mathcal{T}(i)}{\sum_{i=1}^{|\mathcal{X}|}(1 - \mathcal{T}(i))}$$

$$\sum_{i=1}^{|\mathcal{X}|} P(i) = 1$$

En pratique, nous utilisons l'algorithme 19 : nous instancions un tableau de réel *proba* qui contient les probabilités de chaque gène, ainsi, *proba*$[i]$ contient la probabilité du gène i tel qui a été calculé dans la définition 20(Boucle 1 et 2). La somme des éléments du tableau *proba* $= 1$. Nous transformons ce tableau afin de répartir ces probabilités sur une échelle de 0 à 1 dans la boucle 3. Nous pourrons ainsi faire un tirage par roulette proportionnelle tel qui est définit par l'algorithme 20.

L'algorithme roulette décrit notre recherche approchée, nous initialisons le tableau *proba* des probabilités de sélection des gènes. Nous créons une combinaison *combinaison* en fonction de ces probabilités dans la boucle 2 . Si la combinaison caractérise l'instance, nous décrémentons la taille k de la caractérisation souhaitée et nous réinitialisons l'ensemble *combinaison* à l'ensemble vide. Nous réitérons ce processus dans une boucle infinie, nous demandons au programme de s'arrêter à l'issu d'un temps déterminé par l'utilisateur. Notons que la méthode caractérise_instance nous permet de savoir si la combinaison *combinaison* caractérise l'instance est qu'elle bénéficie des heuristiques définit dans la sous-section 2.3.

Remarque 2 (Améliorations de la méthode) *Cette méthode de recherche peut être améliorée : lorsque nous caractérisons une instance avec k gènes, nous pourrions tester les combinaisons possibles de C_k^{k-1} dans la combinaison courante jusqu'à caractériser de nouveau à $k-1$ si cela est possible, et dans le cas*

```
Réel proba[] initialisation_proba (Réel taux[])
// taux[i] est le taux de similarité T du gène i, taux est ordonné par ordre
   croissant de T
Réel sommeTauxInverse ← 0
Réel proba[]
// Boucle 1
pour tout i ∈ [1, ..., |X| faire
|  sommeTauxInverse ← sommeTauxInverse + taux[i]
fin pour
// Boucle 2
pour tout i ∈ [1, ..., |X| faire
|  proba[i] ← (1−taux[i]) / sommeTauxInverse
fin pour
// Boucle 3
pour tout i ∈ [2, ..., |X| faire
|  proba[i] ← proba[i] + proba[i − 1]
fin pour
```

FIGURE 19 – Algorithme d'initialisation des probabilités de sélection des gènes d'une instance

```
roulette (taux, Groupes, k, n)
// taux[i] est le taux de similarité T du gène i, taux est ordonné par ordre
   croissant de T
// Groupes est un ensemble contenant tous les groupes de l'instance
// k correspond au nombre de gène souhaité pour la caractérisation
// n correspond aux nombres de gènes de l'instance
Réel proba[] ← initialisation_proba(taux)
// Boucle 1
tant que VRAI faire
|  Entier combinaison{} // un ensemble d'entier qui correspondra aux indices des
|     gènes dont on souhaite la caractérisation
|  // Boucle 2
|  tant que |combinaison| < k faire
|  |  Réel alea ← nombre aléatoire entre 0 et 1
|  |  // Boucle 3
|  |  pour tout i ∈ [1, ..., I → n] faire
|  |  |  si proba[i] ≥ alea alors
|  |  |  |  combinaison ← combinaison ∪ i
|  |  |  |  Sortir de la boucle 3
|  |  |  fin
|  |  fin pour
|  fin
|  si caractérise_instance(I,combinaison) alors
|  |  afficher("La combinaison ",combinaison," permet de caractériser l'instance.")
|  |  k ← k − 1
|  |  combinaison ← {}
|  fin
fin
```

FIGURE 20 – Algorithme de recherche approchée par roulette proportionelle

contraire uniquement, relancer un tirage. Par manque de temps, nous ne testerons pas cette variante, l'objectif de notre démarche étant de démontrer la pertinence d'utilisation du coefficient $\mathcal{T}$ pour une recherche approchée.

2.4.1 Résultats

Nous présentons ici les résultats de notre méthode. Nous faisons la comparaison avec l'heuristique du programme LSPC (Local Search Proj Car) proposée dans [CLSZ13]. La colonne itération correspond aux nombres d'itérations de la boucle infinie (boucle 1) présentée dans l'algorithme 20.

Instances	Entités	Gènes	$\Delta\mathcal{T}$	ρ	σ	PL	LSPC	Roulette proportionelle		
								k	temps	itérations
s301-0	500	400	0.025	0.034	0.999	-	14	13	421.005	1612515
s326-0	500	500	0.026	0.033	1	-	14	13	93.247	390166
s413-30	500	600	0.027	0.035	1	-	13	13	428.634	1388939
s555-20	800	800	0.029	0.039	0.999	-	13	13	395.024	1286275
s625-20	500	1000	0.027	0.035	1	-	13	13	415.878	1359740
s754-10	600	200	0.027	0.034	1	-	14	14	25.876	84672
s882-20	600	400	0.024	0.032	1	-	14	14	3.023	9113
s2501-70	800	800	0.024	0.033	1	-	15	15	4.24	4350
s31294-50	200	1000	0.049	0.065	0.993	10	11	11	0.963	5273
s3836-0	1000	1000	0.019	0.024	1	-	16	16	5.266	1006
rch8	56	27	0.339	0.569	0.067	**9**	9	9	0.031	22440
raphv	108	68	0.302	0.588	0.419	**6**	9	6	0.657	470528
raphy	112	70	0.294	0.609	0.667	**6**	8	6	0.873	524344
rarep	112	72	0.295	0.651	0.502	**12**	14	14	36.627	17793513
rch10	112	86	0.238	0.626	0.094	**10**	15	12	65.615	35676187

Nous remarquons que notre méthode, bien que grandement améliorable, est sensiblement plus efficace que LSPC sur les instances aléatoires et est beaucoup plus efficace sur les instances réelles : nous trouvons la borne optimale sur les instances rch8, raphv et raphy , et nous l'approchons pour les instances rarep et rch10. Ces résultats démontrent la pertinence de l'utilisation des taux de similarité $\mathcal{T}$ pour les instances réelles.

3 Conclusions et perspectives

Au cours de ce stage, nous avons défini et démontré la pertinence de critères permettant de mesurer la difficulté d'une instance MIN-PCM. Nous avons proposé des heuristiques qui améliorent les temps de résolution et les caractérisations des instances, notamment lorsque celles-ci sont considérées comme étant difficile et ayant la possibilité d'être améliorées.

Nous avons montré que les résolutions des instances aléatoires ne peuvent être améliorées de façon significatives à cause de leurs structures. Cependant, nous sommes désormais en mesure d'outrepasser les difficultés dues à la structure des instances réelles, puisque les résultats obtenus lors de la résolution de celles-ci sont meilleurs que ce qui avait été trouvé jusqu' alors.

Nous constatons qu'une recherche exacte guidée par de bonnes heuristiques ne peut pas prétendre à être utilisée sur des instances de grande taille. En effet, quelle que soit l'heuristique utilisée, nous serons toujours confrontés à l'exhaustivité d'un parcours de C_n^k comparaisons pour prouver l'optimalité d'une solution, et le gain de temps obtenu par une résolution par transformation en problème MIN-ONE se paye par une occupation exponentielle de la mémoire.

Nous obtenons d'excellents résultats avec une recherche approchée. Ces résultats avoisinent la borne optimale des instances. Nous pensons donc qu'il faut privilégier l'axe de la recherche approchée dans nos futurs travaux. A l'avenir, nous aimerions également développer un générateur d'instance difficile qui correspondrait mieux aux structures des instances réelles que manipulent nos collègues en biologie. Ce générateur devra fournir la borne optimale d'une instance lors de sa création afin que nous puissions nous situer lors de leurs résolutions. A plus long terme, nous pourrions nous intéresser aux types de formules renvoyés par nos algorithmes afin que celles-ci puissent fournir des informations aux biologistes(par exemple, $a \vee \neg b \equiv a \Rightarrow b$, ce qui nous indique que la présence du gène a implique la présence du gène b), nous aurions alors un caractère sémantique pour les formules trouvées.

Enfin, il serait intéressant de poursuivre nos recherches avec des biologistes issus d'autres milieux que celui de l'agriculture(la médecine par exemple) afin d'être confrontés à de nouveaux modèles d'instances, présentant de nouvelles spécificités et/ou des difficultés de résolutions plus grandes.

Table des figures

Résumé

Ce rapport porte sur la minimisation du problème de caractérisation multiple (MIN-PCM). Nous dressons un état de l'art, puis nous définissons des éléments permettant de définir qu'une instance est difficile à résoudre ou non. Ces éléments nous permettent aussi de proposer des heuristiques pour des résolutions par recherche exacte et par recherche approchée. Nous présentons les résultats obtenus et faisons la comparaison avec ceux de l'état de l'art.

Mots-clés : MIN-PCM, PCR multiplex, pdbf, heuristiques.

Abstract

This report concerns the minimisation of problem of caracterisation multiple (MIN-PCM). We prepare a state of the art, then we propose elements which can be define the difficult of an instance. These elements also allow us to propose heuristics for resolutions by exact research and approach research. We present the results and make a comparaison with those of the state of art.

Keywords : MIN-PCM, PCR multiplex, pdbf, heuristics.

Références

[BCH+12] Tristan Boureau, Fabien Chhel, Gilles Hunault, Mohamed Kerkoud, Frédéric Lardeux, Charles Manceau, Stéphane Poussier, and Frédéric Saubion. Procédé de depistage de xanthomonas axonopodis pv. phaseoli. Brevet INRA Depose INPI : FR 2970480 (A1), 2012.

[BKC+13] Tristan Boureau, M. Kerkoud, F. Chhel, G. Hunault, A. Darrasse, C. Brin, K. Durand, A. Hajri, S. Poussier, C. Manceau, F. Lardeux, F. Saubion, and M.A. Jacques. A multiplex-PCR assay for identification of the quarantine plant pathogen Xanthomonas axonopodis pv. phaseoli. *Journal of Microbiological Methods*, 92(1) :42–50, 2013.

[CGLS12] Fabien Chhel, A. Goëffon, F. Lardeux, and F. Saubion. Caractérisation multiples minimales utilisant les formules booléennes partiellement définies. *GDRI3IAF*, 2012.

[CLGS12] Fabien Chhel, Frédéric Lardeux, Adrien Goëffon, and Frédéric Saubion. Minimum multiple characterization of biological data using partially defined boolean formulas. In *SAC*, pages 1399–1405. ACM, 2012.

[CLSZ13] Fabien Chhel, F. Lardeux, F. Saubion, and B. Zanuttini. Application du problème de caractérisation multiple à la conception de test de diagnostic pour la biologie végétale. *Revue d'intelligence artificielle*, 27(4-5) :1–20, 2013.

[DV81] P. Descamps and M. Véron. Une méthode de choix des caractères d'identification basée sur le théorème de bayes et la mesure de l'information. *Annal Microbiologie (Paris)*, 132B, 1981.

[GL97] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[HS04] Holger Hoos and Thomas Stützle. *Stochastic Local Search : Foundations & Applications*. Morgan Kaufmann Publishers Inc., 2004.

[MiHI99] Kazuhisa Makino, Ken ichi Hatanaka, and Toshihide Ibaraki. Horn extensions of a partially defined Boolean function. *SIAM Journal on Computing*, 28(6) :2168–2186, 1999.

[SPL+05] N.W. Schaad, E. Postnikova, G.H. Lacy, A. Sechler, I. Agarkova, P.E. Stromberg, V.K. Stromberg, and A.K. Vidaver. Reclassification of xanthomonas campestris pv. citri (ex hasse 1915) dye 1978 forms a, b/c/d, and e as x. smithii subsp. citri (ex hasse) sp. nov. nom. rev. comb. nov., x. fuscans subsp. aurantifolii (ex gabriel 1989) sp. nov. nom. rev. comb. nov., and x. alfalfae subsp. citrumelo (ex riker and jones) gabriel et al., 1989 sp. nov. nom. rev. comb. nov. ; x. campestris pv malvacearum (ex smith 1901) dye 1978 as x. smithii subsp. smithii nov. comb. nov. nom. nov. ; x. campestris pv. alfalfae (ex riker and jones, 1935) dye 1978 as x. alfalfae subsp. alfalfae (ex riker et al., 1935) sp. nov. nom. rev. ; and "var. fuscans" of x. campestris pv. phaseoli (ex smith, 1987) dye 1978 as x. fuscans subsp. fuscans sp. nov. *Systematic Applied Microbiology*, 28(6) :494–518, 2005.

[VHKS95] L. Vauterin, B. Hoste, K. Kersters, and J. Swings. Reclassification of Xanthomonas. *International Journal of Systematic and Evolutionary Microbiology*, 45 :472–489, 1995.