INF580 – Programmation par contraintes

Christoph Dürr *examen écrit final — mars 2014*

Les documents distribués dans le cadre du cours et les notes personnelles sont autorisés. *[english translation at the end of the document]*

1 Variable au plus petit domaine

Une instance d'un problème de satisfaction de contraintes est typiquement résolue en explorant un arbre de recherche, comme vu dans le cours. L'algorithme doit prendre plusieurs décisions qui peuvent influencer sa performance (nombre de noeuds générés dans l'arbre). Par exemple il est possible de choisir comme variable de branchement une variable dont le domaine a la plus petite cardinalité. Qu'est-ce qui motive ce choix, plutôt qu'une variable dont le domaine a la plus grande cardinalité ?

2 Picross

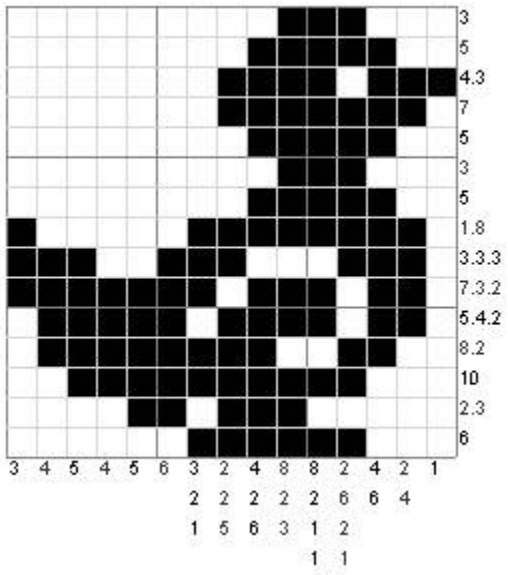

Figure 1: Une instance du problème picross avec sa solution.

Le jeu Picross est moins connu que Sudoku, mais de nature similare. Il consiste en une grille, qu'on supposera de dimension carrée $n \times n$, et d'une liste d'entiers pour chaque ligne et chaque colonne. Nous adoptons la notation suivante.

- $b \in \{0, 1\}$ représente un bit indiquant une colonne $b = 1$ ou une ligne $b = 0$.
- $i \in \{1, \ldots, n\}$ représente un indice de ligne ou de colonne.

- $m_{b,i} \in \{1, \ldots, \lceil n/2 \rceil\}$ est le nombre d'entiers dans la liste associé à la ligne/colonne b, i.
- $a_{b,i,k} \in \{1, \ldots, n\}$ est le k-ème entier de la liste associée à la ligne/colonne b, i pour $k \in \{1, \ldots, m_{b,i}\}$.

Le but est de noircir certaines cases de la grille, tel que pour chaque ligne/colonne b, i, les nombres de cases noirs successifs, forment la liste $a_{b,i,1}, \ldots, a_{b,i,m_{b,i}}$.

Modélisez ce problème, comme un problème de satisfaction de contraintes. Pour chaque variable précisez bien son domaine, indiquez les contraintes qui les lient, et précisez le rôle de chaque contrainte.

3 Arc-consistance

Rendez le problème de satisfaction de contraintes suivant, arc-consistant.

Variables $A \in \{1\}, B \in \{0, 1\}, C \in \{0, 1, 2\}, D \in \{0, 1, 2\}, E \in \{0, 1, 2\}$

Contraintes $A = B$, $B \neq C$, $C \neq D$, $D \neq A$, $C \neq E$, $D \neq E$.

4 Tout différent

Est-ce que l'instance suivante est satisfiable ? Justifiez dans le cas positif par une solution, et dans le cas négatif par un argument.

$$
\begin{aligned}
&A \in \{1, 4\} \\
&B \in \{1, 2, 5, 7\} \\
&C \in \{1, 3, 4\} \\
&D \in \{2, 4, 5\} \\
&E \in \{3, 4\} \\
&F \in \{2, 3, 5, 6\} \\
&G \in \{3, 4\} \\
&\quad \text{AllDifferent}(A, B, C, D, E, F, G)
\end{aligned}
$$

5 Chocolat: Modélisation en programmation linéaire à variables entières

On veut imprimer des emballages pour des chocolats. Il existe trois sortes de chocolats : chocolat au lait, chocolat noir et chocolat aux amandes. Pour éviter des pertes exactement

4 emballages peuvent être imprimés sur une même planche. Imprimer k planches identiques coûte (10+ k) Euros. Pour les trois sortes de chocolats nous avons besoin de respectivement 120, 250, 130 emballages. Notre unique but est de minimiser les frais d'impression, quitte à gaspiller des emballages inutilisés.

Modélisez ce problème par un programme linéaire à variables entières, en commentant le sens des variables et des contraintes.

6 Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2: Une instance du problème sudoku.

On ne présente plus le problème de Sudoku. Une instance est donnée par une grille 9×9 avec certaines cases préremplies par des entiers entre 1 et 9. La grille comporte 9 ligne, 9 colonnes et 9 blocks, où un bloc est une sous-grille 3×3, et la cellule en ligne i et colonne j appartient au k-ème bloc avec

$$k = 1 + \lfloor (j-1)/3 \rfloor + 3\lfloor (i-1)/3 \rfloor.$$

Le but est de compléter la grille, tel que dans chaque ligne, colonne ou bloc, chaque valeur de 1 à 9 apparaisse exactement une fois.

Modélisez le problème de Sudoku, comme un problème de satisfiabilité. Commentez la forme de la formule ainsi obtenue, par rapport à la classification du théorème de Schaeffer.

7 Résolution

Voici une formule booléenne SAT. Chaque ligne représente une clause.

$$
\begin{aligned}
&x_1 \vee x_2\\
&x_1 \vee \neg x_2 \vee x_4\\
&x_3 \vee \neg x_4 \vee x_5 \vee \neg x_6\\
&x_3 \vee \neg x_4 \vee x_5 \vee x_6\\
&x_3 \vee \neg x_5 \vee x_7\\
&x_3 \vee \neg x_5 \vee \neg x_7\\
&\neg x_3 \vee \neg x_4 \vee x_8\\
&\neg x_3 \vee \neg x_4 \vee \neg x_8\\
&\neg x_1 \vee \neg x_2\\
&\neg x_1 \vee x_2 \vee \neg x_3\\
&x_2 \vee x_3 \vee \neg x_4\\
&\neg x_1 \vee x_4 \vee x_8\\
&\neg x_1 \vee x_4 \vee \neg x_8
\end{aligned}
$$

1. Appliquez la résolution (règle 6 du cours) sur les variables dans l'ordre x_8, x_7, x_6, x_5, x_4 et simplifiez après chaque étape. Précisez à chaque étape quelles règles vous appliquez.

2. Donnez une assignation valide de la formule ou indiquez que la formule n'est pas satisfiable.

8 Recherche locale

Rahul veut tourner un clip vidéo dans la ville de Vuong. Pour cela il cherche un carrefour et une période dans la journée particulièrement calme. Il sait bien que cette tâche est difficile alors il se contente d'un *minimum local*, concrètement un carrefour qui n'est pas plus bruyant que les carrefours voisins pendant cette période, et qui n'est pas plus bruyant que dans la période précédente ou suivante. La ville Vuong est particulièrement régulière, et est composée de n rues nord-sud et de n rues ouest-est, formant une grille de n^2 carrefours. Une journée est divisée en n périodes. Et la vie dans cette ville est tellement monotone que le niveau sonore d'un carrefour est le même dans une même période, d'un jour à l'autre. Rahul dispose d'une camionette qui lui permet de se déplacer et de mesurer les niveaux sonores. Ce sont les mesures qui lui coûtent de l'argent, les déplacements sont gratuits, et il n'a pas de limite de temps.

Bref, Rahul dispose d'une matrice 3-dimensionelle entière M de dimension $n \times n \times n$, et pour éviter les effets de bords on suppose $M_{ijk} = +\infty$ quand un des indices i, j, k est 1 ou

n. Rahul y cherche un minimum local, donc un triplet $2 \leq i, j, k \leq n-1$ tel que

$$M_{ijk} \leq \begin{cases} M_{i-1,j,k} \\ M_{i+1,j,k} \\ M_{i,j-1,k} \\ M_{i,j+1,k} \\ M_{i,j,k-1} \\ M_{i,j,k+1} \end{cases}$$

Analysez la complexité dans le pire des cas d'une recherche locale basée sur

1. la descente locale
2. diviser et conquérir.

9 Trier

Il paraît que la programmation linéaire à variable entières est un outil algorithmique très général. Testons ceci sur le problème du tri. Étant donnée une liste de n entiers $w_1, \ldots, w_n$ — qu'on suppose tous distincts pour rendre la solution unique — on veut les ordonner en ordre croissant.

1. Proposez une formulation par programme linéaire à variables entières. Explicitez la correspondance entre une solutions entière un ordre sur la liste donnée en entrée.
2. Est-ce que le polytope décrit par le programme linéaire relâché n'a que des sommets entiers? Dans le cas positif donnez une preuve, dans le cas négatif un contre exemple.

10 Ensemble distant

Étant donné un graphe $G(V, E)$, la distance entre deux sommets distincts $u, v \in V$ est la longueur du plus court chemin reliant u à v. Si les deux sommets ne sont pas connectés, la distance est définie comme $+\infty$, et s'il existe une arête (u, v) la distance est 1.

Étant donnée un graphe $G(V, E)$ et un entier k, on dit qu'un ensemble $S \subseteq V$ est k-distant si la distance entre tout couple $u, v \in S$ avec $u \neq v$ est au moins k. On veut trouver un tel ensemble de cardinalité maximale. Pour cela on cherche un algorithme exponentiel en $|V|$.

1. Donnez un algorithme polynomial pour le cas où chaque sommet a au plus deux voisins.
2. Décrivez algorithme dans le cas général et analysez sa complexité. Il suffit de donner la récursion linéaire de la complexité sous la forme

   $$T(n) \leq T(n - a_1) + \ldots + T(n - a_\ell) + O(n^c)$$

   pour une séquence $a_1, \ldots, a_\ell$ et une constante c.

A English translation

A.1 Variable with smallest domain

An instance to the satisfaction programming problem is typically solved by exploring a search tree, as seen in the course. The algorithm has to make several decisions which can influence its performance (number of generated nodes in the tree). For example it is possible to choose as branching variable, a variable with a domain of smallest cardinality. What motivates this choice, rather than a variable with a domain of largest cardinality?

A.2 Picross

The game Picross is less known than Sudoku, but of same nature. It consists of a grid, which we assume of squared dimension $n \times n$ and a list of integers for each row and each column (see Figure). We use the following notation.

- $b \in \{0, 1\}$ represents a bit, indicating a column $b = 1$ or a row $b = 0$.
- $i \in \{1, \ldots, n\}$ represents a column or row index.
- $m_{b,i} \in \{1, \ldots, \lceil n/2 \rceil\}$ is the number of integers in the list associated to the row/column b, i.
- $a_{b,i,k} \in \{1, \ldots, n\}$ is the k-th integer of the list associated to the row/column b, i for $k \in \{1, \ldots, m_{b,i}\}$.

The goal is to darken some cells of the grid, such that in each row/column b, i, the number of successive black cells, form the list $a_{b,i,1}, \ldots, a_{b,i,m_{b,i}}$.

Model this problem as a constraint satisfaction problem. For every variable indicate its domain, and define the constraints linking the variables. Explain the role of each constraint.

A.3 Arc-consistancy

Make the following CSP arc-consistant.

Variables $A \in \{1\}, B \in \{0, 1\}, C \in \{0, 1, 2\}, D \in \{0, 1, 2\}, E \in \{0, 1, 2\}$

Constraints $A = B$, $B \neq C$, $C \neq D$, $D \neq A$, $C \neq E$, $D \neq E$.

A.4 All different

Is the following CSP instance feasible? In the positive case provide a solution, while in the negative case provide an argument.

$$
\begin{aligned}
&A \in \{1, 4\} \\
&B \in \{1, 2, 5, 7\} \\
&C \in \{1, 3, 4\} \\
&D \in \{2, 4, 5\} \\
&E \in \{3, 4\} \\
&F \in \{2, 3, 5, 6\} \\
&G \in \{3, 4\} \\
&\quad \text{AllDifferent}(A, B, C, D, E, F, G)
\end{aligned}
$$

A.5 Chocolate: Modelizing in mixed integer linear programming

We want to print boxes for chocolate. There are 3 types of chocolate: milk chocolate, dark chocolate and almond chocolate. To reduce cost we want to print exactly 4 boxes on each cardboard. Printing k identical cardboards costs (10+ k) Euros. For the 3 types of chocolates we need respectively 120, 250, 130 boxes. Our unique goal is to minimize the printing costs, allowing to throw away unused boxes.

Model this problem with a mixed integer linear program, commenting the meaning of the variables and constraints.

A.6 Sudoku

We do not present the Sudoku problem anymore. An instance is given by a 9 by 9 grid, with some cells already filled by integers taken from $1, \ldots, 9$. The grid consists of 9 rows, 9 columns and 9 blocs, where a block is a 3 by 3 subgrid, and the cell in row i and column j belongs to block k with

$$k = 1 + \lfloor (j-1)/3 \rfloor + 3\lfloor (i-1)/3 \rfloor.$$

The goal is to complete the grid, such that in every row, column and block, every value between 1 and 9 appears exactly once.

Model the Sudoku problem as a satisfiability problem. Comment the form of the obtained formula with respect to the classification in Schaeffer's theorem.

A.7 Resolution

Here is a boolean SAT formula. Every row represents a clause.

$$
\begin{aligned}
&x_1 \vee x_2 \\
&x_1 \vee \neg x_2 \vee x_4 \\
&x_3 \vee \neg x_4 \vee x_5 \vee \neg x_6 \\
&x_3 \vee \neg x_4 \vee x_5 \vee x_6 \\
&x_3 \vee \neg x_5 \vee x_7 \\
&x_3 \vee \neg x_5 \vee \neg x_7 \\
&\neg x_3 \vee \neg x_4 \vee x_8 \\
&\neg x_3 \vee \neg x_4 \vee \neg x_8 \\
&\neg x_1 \vee \neg x_2 \\
&\neg x_1 \vee x_2 \vee \neg x_3 \\
&x_2 \vee x_3 \vee \neg x_4 \\
&\neg x_1 \vee x_4 \vee x_8 \\
&\neg x_1 \vee x_4 \vee \neg x_8
\end{aligned}
$$

1. Apply resolution (rule 6 in the course) on the variables in the order x_8, x_7, x_6, x_5, x_4 and simplify after each step. Indicate at each step which rules you applied.

2. Give an assignment validating the formula or indicate that the formula is not satisfiable.

A.8 Local search

[we removed to introductory story, to keep it short]

We are given a 3-dimensional integer matrix M of dimension $n \times n \times n$, and to avoid dealing with the border we assume $M_{ijk} = +\infty$ whenever one of the indices i, j, k is 1 or n. We want to find a local minimum, i.e. a triplet $2 \leq i, j, k \leq n-1$ such that

$$
M_{ijk} \leq \begin{cases} M_{i-1,j,k} \\ M_{i+1,j,k} \\ M_{i,j-1,k} \\ M_{i,j+1,k} \\ M_{i,j,k-1} \\ M_{i,j,k+1} \end{cases}
$$

Analyze the worst case complexity of local search based on

1. local descend

2. divide and conquer.

A.9 Sorting

People say that mixed integer linear programming is a very general algorithmic tool. Let's try it on the sorting problem. We are given a sequence of n integers $w_1, \ldots, w_n$ — which we suppose all different, to make the solution unique — and the goal is to rearrange them in increasing order.

1. Propose a mixed integer linear programming formulation of this problem. Explain the correspondance between an integer solution and an order on the input sequence.

2. Does the polytope generated by the relaxed linear program have only integer extrem point vertices? In the positive case give a proof, while in the negative case give an example.

A.10 Distant set

Given a graph $G(V, E)$, the distance between two distinct vertices $u, v \in V$ is defined as the length of the shortest path connecting them. If there is no such path, the distance is defined as ∞. If there is an edge (u, v) the distance is 1.

A set $S \subseteq V$ is called k-distant if every pair of vertices $u, v \in S$ with $u \neq v$ has distance at least k. Given $G(V, E)$ and an integer k, the goal is to find a k-distant set of maximum cardinality. For this we aim for an exponential algorithm in $|V|$.

1. Give a polynomial time algorithm in the case every vertex has at most 2 neighbors.

2. Describe an algorithm for the general case and analyze its complexity. It is enough to provide the linear recurrence of its complexity in the form

$$T(n) \leq T(n - a_1) + \ldots + T(n - a_\ell) + O(n^c)$$

for some sequence $a_1, \ldots, a_\ell$ and constant c.

B Corrections

B.1 Variable au plus petit domaine

Dans une optique d'obtenir un petit arbre de recherche, on aimerait réduire le degré des noeuds. Bien sûr il n'y a pas de garantie pour une instance fixée, que ce choix est mieux qu'un autre, c'est pourquoi on parle d'*heuristique*.

B.2 Picross

Il nous faut deux types de variables. D'une part, nous devons indiquer pour chaque b, i, k, la position de k-ème barre dans la ligne/colonne b, i. Et d'autre part nous devons indiquer pour chaque case de la grille, si elle noircie ou pas. Si seulement un des ces deux types de variables suffit pour décrire une solution, les deux semblent necessaire pour imposer toutes les contraintes.

Variables Pour tout $b \in \{0,1\}$, $i \in \{1,\dots,n\}$ et $k \in \{1,\dots,m_k\}$ nous avons la variable $p_{b,i,k} \in \{1,\dots,n\}$. Et pour tout $i,j \in \{1,\dots,n\}$ nous avons la variable $M_{i,j} \in \{0,1\}$. Pour les variables $M_{i,j}$, la valeur 0 représente la couleur blanche et 1 la couleur noire.

Contrainte "ordre" Pour tout b, i et $k \in \{1,\dots,m_{b,i}-1\}$, $p_{b,i,k}+a_{b,i,k}+1 \leq p_{b,i,k+1}$. Cette contrainte assure que les barres sont dans l'ordre demandé dans chaque ligne/colonne b, i et séparés par au moins une case blanche, d'où le $+1$.

Contrainte "noir" Pour tout $b \in \{0,1\}$, $i,j \in \{1,\dots,n\}$ et $k \in \{1,\dots,m_{b,i}\}$, nous imposons $j < p_{b,i,k}$ ou $j \geq p_{b,i,k} + a_{b,i,k}$ ou $b = 0$ et $M_{ij} = 1$ ou $b = 1$ et $M_{ji} = 1$. Dans cette expression le *et logique* a priorité devant le *ou logique*. Cette contrainte assure que les cellules couvertes par des barres sont tous noir.

Contrainte "blanche" Pour tout i, $\sum_j M_{i,j} = \sum_{k=1}^{m_{0,i}} a_{0,i,k}$ et $\sum_j M_{j,i} = \sum_{k=1}^{m_{1,i}} a_{1,i,k}$. Cette contrainte assure que toutes les cases noires sont couvertes par une barre.

Suivant le solveur CSP utilisé, il sera peut-être possible d'introduire des variables intermédiaires pour coder les additions dans les indices. Dans ce cas on demandera au solveur de ne pas brancher sur ces variables là.

B.3 arc-consistance

Variables $A \in \{1\}, B \in \{1\}, C \in \{0,2\}, D \in \{0,2\}, E \in \{0,1,2\}$.

B.4 Tout différent

Les variables A, C, E, G ont leur domaine inclus dans $\{1,3,4\}$. La condition de Hall est donc violée, 4 variables ne peuvent pas prendre des valeur distincts dans un domaine de taille 3, il n'y a donc pas de solution.

B.5 Chocolat

Soit S l'ensemble de tous les triplets (l, n, a) tel que $l, n, a \geq 0$ et $l + n + a = 4$. Chaque triplet (l, n, a) représente une planche avec l emballages de chocolat au lait, n emballages de chocolat noir et a emballages de chocolat aux amandes. Il n'y a que 10 tels triplets, car il y a un triplet avec $l = 0$, deux avec $l = 1$, trois avec $l = 2$ etc.

Pour chaque triplet, il faut décider si on imprime de tels planches, c'est le rôle des variables P et dans le cas positif, combien, ce qui est le rôle des variables Q.

$$\begin{aligned}
&\min \sum_{lna \in S} (10P_{lna} + Q_{lna}) \\
\text{s.t.} &\forall lna \in S : Q_{lna} \leq MP_{lna} \\
&\sum_{lna \in S} lQ_{lna} \geq 120 \\
&\sum_{lna \in S} nQ_{lna} \geq 250 \\
&\sum_{lna \in S} aQ_{lna} \geq 130 \\
&\forall lna \in S : Q_{lna} \in \mathbb{N} \\
&\forall lna \in S : P_{lna} \in \{0, 1\}
\end{aligned}$$

Ici M est une constante suffisamment grande, par exemple $\max\{120, 250, 130\}$.

B.6 Sudoku

Pour chaque cellule p et valeur v, notre modèle introduit une variable booléenne X_{pv}, qui indique si la cellule contient la valeur v. Pour chaque couple p, v donné dans la grille initiale, la variable correspondante est posée à *vrai*.

Pour toute valeur v et tout couple de cellules $p \neq q$ dans une même ligne, colonne ou bloc, on a la contrainte $\overline{X_{pv}} \vee \overline{X_{qv}}$, assurant qu'une valeur n'y apparait pas deux fois.

Pour toute cellule p et chaque couple de valeurs $u \neq v$, on a la contrainte $\overline{X_{pu}} \vee \overline{X_{pv}}$, assurant qu'au plus une valeur est inscrite dans chaque cellule.

Jusqu'a maintenant la formule est de la forme 2-SAT, mais il faut aussi imposer que chaque cellule contient au moins une valeur $X_{p1} \vee \ldots \vee X_{p9}$, ce qui en fait une formule SAT, qui n'a pas une des formes pour lesquelles un algorithme polynomial est connu. Ceci dit toutes les formules ainsi obtenues ont une taille constante, et donc le temps de résolution est constant.

B.7 Résolution

Nous utilisons pour la correction une notation compacte où par exemple $1\bar{2}4$ représente la clause $x_1 \vee \overline{x_2} \vee x_4$.

donnée	12	$1\bar{2}4$	$3\bar{4}5\bar{6}$	$3\bar{3}56$	$3\bar{5}7$	$3\bar{5}\bar{7}$	$\bar{3}\bar{4}8$	$\bar{3}\bar{4}\bar{8}$	$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$	$23\bar{4}$	$\bar{1}48$	$\bar{1}4\bar{8}$
rés.8	12	$1\bar{2}4$	$3\bar{4}5\bar{6}$	$3\bar{3}56$	$3\bar{5}7$	$3\bar{5}\bar{7}$	$\bar{3}\bar{4}\bar{3}\bar{4}$	$\bar{3}\bar{4}\bar{1}4$	$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$	$23\bar{4}$	$\bar{1}4\bar{3}\bar{4}$	$\bar{1}4\bar{1}4$
règle 1	12	$1\bar{2}4$	$3\bar{4}5\bar{6}$	$3\bar{3}56$	$3\bar{5}7$	$3\bar{5}\bar{7}$	$\bar{3}\bar{4}$		$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$	$23\bar{4}$		$\bar{1}4$
rés.7	12	$1\bar{2}4$	$3\bar{4}5\bar{6}$	$3\bar{3}56$	$3\bar{5}$		$\bar{3}\bar{4}$		$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$	$23\bar{4}$		$\bar{1}4$
rés.6	12	$1\bar{2}4$	$3\bar{4}5$		$3\bar{5}$		$\bar{3}\bar{4}$		$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$	$23\bar{4}$		$\bar{1}4$
rés.5	12	$1\bar{2}4$	$3\bar{4}$				$\bar{3}\bar{4}$		$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$	$23\bar{4}$		$\bar{1}4$
règle 2	12	$1\bar{2}4$	$3\bar{4}$				$\bar{3}\bar{4}$		$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$			$\bar{1}4$
rés.4	12	$1\bar{2}3$	$1\bar{2}\bar{3}$				$\bar{1}3$		$\bar{1}\bar{2}$	$\bar{1}2\bar{3}$			$\bar{1}\bar{3}$
règle 2	12	$1\bar{2}3$	$1\bar{2}\bar{3}$				$\bar{1}3$		$\bar{1}\bar{2}$				$\bar{1}\bar{3}$
rés.3	12	$1\bar{2}$	$1\bar{2}\bar{1}$				$\bar{1}1\bar{2}$		$\bar{1}\bar{2}$				$\bar{1}$
règles 1,2	12	$1\bar{2}$											$\bar{1}$
rés.2	1												$\bar{1}$

Finalement la résolution sur x_1, suivi de la règle 3, montre que la formule n'est pas satisfiable.

B.8 Recherche locale

La descente locale peut prendre un temps $\Omega(n^3)$. Imaginons que n soit impair, alors en plaçant ∞ dans certaines cases de la matrice, on peut laisser la place à un unique chemin que la descente locale suivra et dont la longueur est $\Theta(n^3)$. L'approche diviser et conquérir (voir Figure 3) a une complexité de

$$T(n) \leq n^2 + n(n/2) + (n/2)^2 + T(n/2) + O(1)$$

ce qui donne $T(n) = O(n^2)$.

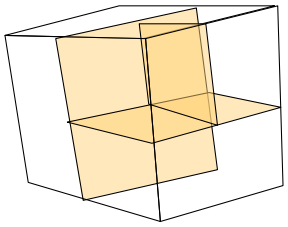

Figure 3: Les étapes de l'approche diviser et conquérir

B.9 Tri

Nous introduisons les variables X_{ir} qui valent 1 si dans la solution l'entier w_i est en rang r. Les contraintes sont $\forall i : \sum_r X_{ir} = 1$ et $\forall r : \sum_i X_{ir} = 1$. Ce programme linéaire

modélise le problème du couplage parfait dans un graphe bi-parti complet, et est donc entier. Alternativement, toute solution fractionnelle X a la propriété que pour chaque valeur fractionnelle, il existe au moins une autre dans la même ligne et la même colonne. Il est alors possible de construire un cycle alternant, qui permet de prouver que X n'est pas une solution point extrème. La fonction objective $\min \sum_{i,r} rX_{ir}w_i$ assure par un argument d'échange que $w_i < w_j$ si et seulement si, dans la solution le rang de w_i est avant celui de w_j.

B.10 Ensemble distant

Notez, le cas particulier $k = 2$ est le problème NP-dur d'ensemble indépendant maximum dans un graphe.

Si chaque sommet a au plus deux voisins, le graphe consiste en une collection de chemins et cycles sommet disjoints. On peut résoudre chaque composante indépendamment. La solution optimale pour un chemin de p sommets est $\lceil p/k \rceil$ et $\lfloor p/k \rfloor$ pour un cycle de p sommets. Un parcours en profondeur d'abord (DFS) résout dans ce cas le problème en temps linéaire.

Soit $N(v)$ l'ensemble des sommets u de distance inférieure à k avec v. Par définition $v \in N(v)$.

Dans le cas général, on choisit un sommet v maximisant $|N(v)|$. On a la promesse que $|N(v)| \geq \max\{4, k+1\}$.

La solution ne contient soit pas v, soit elle contient v, mais alors pas ses vo. Elle est donc le maximum entre $OPT(G \backslash \{v\})$ et $1 + OPT(G \backslash N(v))$, où $N(v)$ est le voisinage de v inclus. La complexité $T(n)$ pour $n = |V|$ est alors constante pour $n \leq 1$ et

$$T(n) \leq T(n-1) + T(n - \max\{4, k+1\}) + O(n)$$

sinon. Le dernier terme linéaire couvre à la fois la recherche du sommet de plus grand degré ou le cas polynomial cité ci-haut. On suppose que le graphe est donnée sous forme de liste d'adjacence.