PROJET SDD L2 2016
*C****z* : A Sliding Puzzles Framework
v0.4

Franck Lepoivre

8 novembre 2012

Seconde version pour questions, commentaires et demandes de précisions. Cette version vous permet de commencer votre travail de documentation et de réflexion. Elle est susceptible de très nombreuses évolutions d'ici la fin du mois d'octobre, notamment en fonction de nos interactions.

1 Résumé

Le sujet porte sur les *puzzles à pièces coulissantes* également appelés *casse-tête de déplacements*. Il s'agit d'identifier les plus difficiles d'entre-eux.

Figure 1 – Dad's Puzzle de 1926

Il vous faudra comprendre en profondeur les principes de cette famille très large de casse-tête, en prenant le temps d'effectuer un travail de collecte et de synthèse d'information.

Vous serez alors en mesure d'élaborer un modèle générique pour représenter de tels jeux ; c'est ce modèle qui vous servira d'appui pour concevoir et réaliser SDD et algorithmes réutilisables pour chacun d'entre-eux.

La réalisation de cet enjeu de réutilisation prendra la forme d'un canevas applicatif (librairie, fichiers de paramétrage, emplacements pour intégrer les éléments de déclinaisons applicatives, documentation pour le développeur) et d'un jeu d'au moins quatre applications, dont deux figures imposées et deux figures de votre choix.

2 Sujet

2.1 But

Le but est de vous placer sur les traces de John H. Conway qui en 1975, avec Century, une évolution de l'Âne Rouge, ouvrit la voix à la recherche des puzzles les plus difficiles composés d'un minimum de pièces. En l'occurrence, il s'agissait d'effectuer la recherche systématique pour trouver le puzzle à pièces coulissantes rectangulaires disposées sur un plateau de 4×5 le plus difficile possible (étude publiée dans son ouvrage fondateur de la *théorie combinatoire des jeux*, *Winning Ways for your Mathematical Plays* [1])

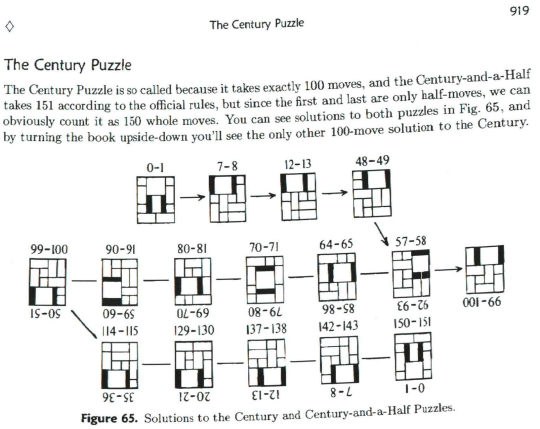

Figure 65. Solutions to the Century and Century-and-a-Half Puzzles.

FIGURE 2 – Solution du Century de John Conway (Extrait de *Winning Ways*)

2.2 Objectifs pédagogiques

Le but fonctionnel du projet vous place dans une situation d'ingénierie qui vous oblige à :

Réfléchir aux fondamentaux d'une famille d'applications (puzzles à pièces coulissantes) pour en dégager un modèle général applicable aux jeux existants (Taquin, Âne Rouge, etc) mais aussi à ceux restant à inventer.

Traduire ce modèle général en structures de données et algorithmes réutilisables.

Appliquer ce canevas à la réalisation de plusieurs applications, dont deux figures imposées, une figure existante au choix, une nouveauté.

1. L'étude des puzzles de déplacement est donnée dans le chapitre 24 *Pursuing Puzzles Purposefully*, section *A Trio of Sliding Block Puzzles*, pp 877 - 884 puis dans les *Extras* du même chapitre, sous-section *The Century Puzzle*, p 919 dont la copie facsimilé est donnée en illustration ci-dessus.

3 Les puzzles à pièces coulissantes

Les *puzzles à pièces coulissantes* font leur apparition à la fin du 19e siècle avec le désormais célèbre Taquin (en anglais, le *fifteen puzzle*) inventé par Noyes Chapman en 1880 et popularisé par le joueur d'échecs Sam Loyd. Ce dernier s'en attribua la paternité, dans son ouvrage de référence, *Cyclopedia of 5000 Puzzles*.

Ces puzzles sont constitués d'un ensemble de pièces plates, positionnées sur un même *plateau* de jeu dont elles ne recouvrent pas totalement la surface (il reste un ou plusieurs *vides*).

Bien qu'elles se contraignent mutuellement, ces pièces sont mobiles par glissement (il est interdit de les soulever pour les retirer, les superposer, etc). En effet, la présence de vide permet, depuis tout arrangement des pièces (qu'on appelle *configuration* ou *état de jeu*), de faire glisser une ou plusieurs pièces vers un vide pour produire une nouvelle configuration.

Partant d'une configuration initiale, le but du jeu est de faire apparaître une configuration particulière de fin de jeu (pas nécessairement unique) en suivant un enchaînement d'étapes, chacune représentée par un mouvement élémentaire. On appelle cet enchaînement *chemin de résolution*. Il existe naturellement un chemin de longueur minimale, le *chemin optimal*.

3.1 L'ancêtre de tous : le Taquin

Le *Taquin* est composé de pièces carrées, de même taille, qui recouvrent ensemble une surface carrée dans laquelle est maintenue une *place vide* de la taille d'une pièce. La place vide assure qu'à tout moment, un mouvement est possible.

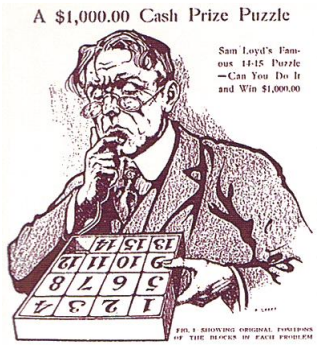

FIGURE 3 – Le Taquin par son usurpateur

La fin de partie est caractérisée par une configuration particulière des pièces (par exemple, la recomposition d'une image).

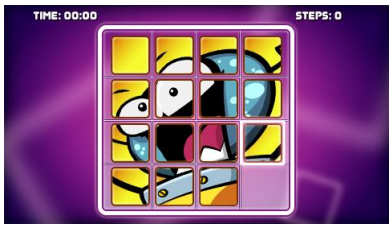

FIGURE 4 – Un iTaquin

3.2 L'Âne Rouge : irrégularité des pièces ⇒ difficulté

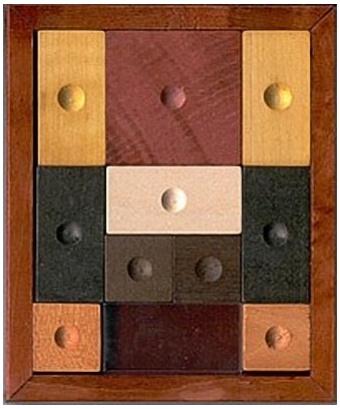

FIGURE 5 – Âne rouge en bois

Héritier du Taquin, l'Âne Rouge [2] est apparu dans la même période. Avec seulement dix pièces, il peut donner l'impression d'être un jeu simple. Le chemin de résolution optimal comporte pourtant 80 étapes, dont voici les majeures (à vous de découvrir les étapes mineures qui relient les majeures entre elles) :

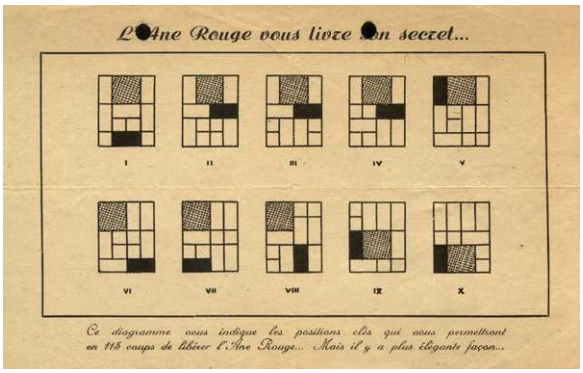

FIGURE 6 – Guide de résolution de l'Âne Rouge

L'évolution du Taquin qui consiste à varier les formes et tailles des pièces du puzzle augmente radicalement la difficulté du casse-tête.

Dans cet environnement fortement contraint, les longues chaînes de mouvements simples à effectuer ne sont pas sans rappeler celles qui conduisent à résoudre le problème des Tours de Hanoï.

La voie est ouverte pour la recherche de jeux de casse-tête basés sur le principe des pièces coulissantes composés d'un minimum de pièces mais qui obligent le joueur à un nombre minimum d'étapes important avant de parvenir à une solution.

2. couramment appelé Klotski par les anglo-saxons, d'après le Polonais *bloc de bois*

3.3 La généralisation des puzzles sur plateau 4×5

Conservons le même plateau, mais varions les combinaisons de pièces et les configurations initiales : le *Dad's puzzler* de 1909 est en bois, ce qui ne permet pas de faire varier les dimensions de son plateau : ceci n'est plus vrai avec les applications informatiques qui permettent de jouer également sur cet élément :

`http://www.proprofs.com/games/sliding-block-puzzle`
`http://www.coolmath-games.com/0-slidingblocks`

3.4 Le plus petit puzzle le plus difficile à résoudre

Avec Century, en 1975, Conwey lance la recherche systématique du jeu le plus difficile sur une grille 4×5, en utilisant les mêmes pièces que celles de l'Âne Rouge.

L'idée est de trouver le plus simple puzzle à pièces coulissantes qui soit aussi le plus difficile à résoudre. Plus formellement, il s'agit de trouver un chemin optimal de résolution qui soit le plus long possible.

Cette idée de recherche systématique, notamment assistée d'un ordinateur, a été reprise par Quzzle [**DEVELOPPER**].

4 Cahier des charges

4.1 Vocabulaire

Fixons le vocabulaire de référence qui désignera les objets matériels et conceptuels manipulés par le canevas et les applications à réaliser.

Plateau : support plan sur lequel prend place le jeu, dans le cadre de la grille.

Grille : division du plateau en places, de même forme (pavage régulier à base de polygones), qui contraint les possibilités de positionnement des pièces du puzzle.

Pièce : bloc plein polygonal qui recouvre un ensemble de places adjacentes.

4.2 Concevoir et réaliser le canevas

La réalisation de cet enjeu de réutilisation prendra la forme d'un canevas applicatif (librairie, fichiers de paramétrage, emplacements pour intégrer les éléments de déclinaisons applicatives, documentation pour le développeur) et d'un jeu d'au moins quatre applications, dont deux figures imposées et deux figures de votre choix.

4.2.1 Le jeu type

Un puzzle à pièces coulissantes qui n'est pas nécessairement enfermé dans le modèle classique à base de grille plane à maille carrée. Cela constituera pourtant le modèle de base attendu.

Modèle de base Le *plateau* de jeu est décrit d'une part par une *grille* à *maille* carrée que l'on appellera le *carré élémentaire*, chaque carré élémentaire définissant une *place*.

D'autre part, le plateau est délimité par une *bordure* qui forme la frontière entre un en-dedans et un en-dehors. La bordure forme un rectangle composé d'un nombre entier de places. On dira que le plateau est $n \times m$ pour dire qu'il est constitué de n rangées (horizontales) de m places.

De même, les pièces du jeu seront des rectangles multiples du carré élémentaire et dont les deux dimensions sont respectivement majorées par n et m.

Extensions du modèle de base La première extension consiste à ne plus limiter la forme du plateau, ni celle des pièces, à des formes rectangulaires.

Il est possible d'envisager une grille non carrée :

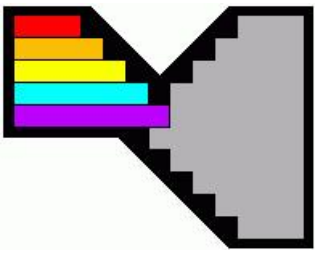

FIGURE 7 – Junk's Hanoi by Junk Kato

De même, les pièces peuvent prendre les formes de celles du jeu bien connu *Tetris* (les sept tétrominos).

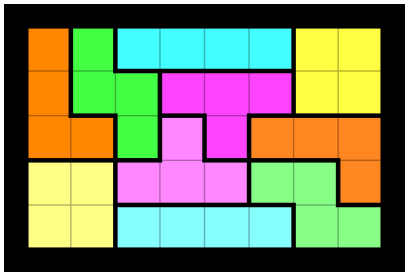

FIGURE 8 – Tétrominos de Tétris

Les plus imaginatifs pourront envisager un plateau sans frontières (torique), le passage du 2D au 3D (tétraminos), des mailles non carrées (est-ce seulement possible ?), etc.

4.2.2 La partie type

- Charger la position initiale de jeu (à partir d'un fichier de configuration)
- Tant que la partie n'est pas terminée :

  Chrono chronométrer le joueur et compter le nombre de mouvements joués.

  Jeu Permettre de jouer l'un des coups possibles (glissement d'une pièce vers une position vide). Cela implique de gérer les transitions d'états. En option, permettre d'annuler les derniers mouvements et de rejouer un certain nombre de ceux qui viennent d'être annulés (`Ctrl + Z` et `Ctrl + Y`).

  Reste à faire Indiquer à quelle distance de la position finale est située la position courante (si possible, par le plus court chemin).

  Aide Indiquer la position (l'objectif) à 10 coups de la position courante, sur le chemin de résolution optimal.

4.3 Module supplémentaire d'aide à la conception d'applications

Pour une application définie par un plateau et un ensemble de pièces :

Vérification de cohérence Vérifier la validité et la cohérence du plateau et de l'ensemble de pièces. Être notamment en mesure de rejeter un jeu de pièces qui ne pourrait être placé sur la plateau en aucune manière, ou encore qui laisserait trop de vide(s) pour permettre de contraindre suffisamment les possibilités de mouvements respectifs des pièces les unes par rapport aux autres. Vous préciserez d'ailleurs ces notions dans le cadre de l'étude présentée dans votre rapport.

Analyse des possibilités Analyser l'espace d'états : nombre d'états, de transitions, composantes connexes, facteur de branchement.

Recherche des jeux difficiles Rechercher les meilleurs couples d'états c'est-à-dire ceux qui sont reliés par le chemin optimal le plus long possible [3].

Guide de résolution Pour deux états donnés, l'un de départ, l'autre d'arrivée, affichage du plus court chemin sous forme d'enchaînement de mouvements à effectuer [4].

3. Rappel : le but du projet est la recherche systématique des jeux les plus difficiles possibles
4. format de base donné en annexe qui peut être complété (mais pas remplacé) par un visuel plus élaboré

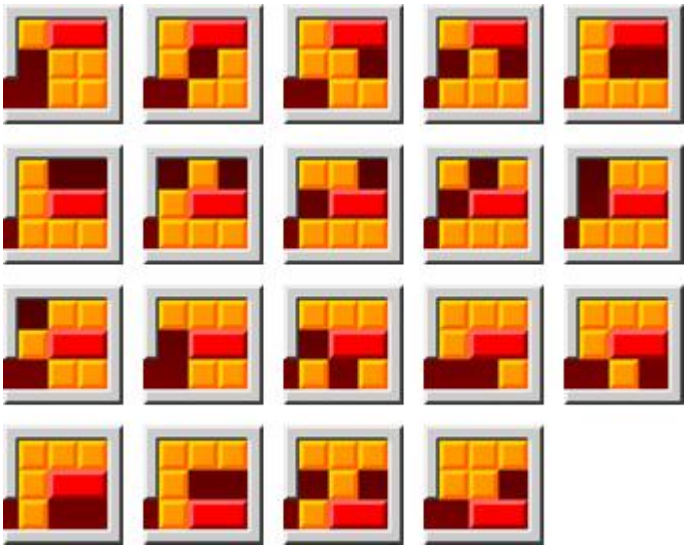

FIGURE 9 – Exemple de guide de résolution

4.4 Mettre en application

Réaliser au moins quatre programmes en utilisant les éléments communs du canevas :

Le Taquin de dimensions rectangulaires de taille quelconque (ceci doit pouvoir être paramétré).

L'Âne rouge ...

Un ou plusieurs jeux existants de votre choix.

Un ou plusieurs jeux qui n'existent pas encore : vous en serez les inventeurs.

5 Organisation

5.1 Encadrement et calendrier

Cette année, le projet de SDD est encadré sur la durée de son exécution, et non pas seulement évalué en décembre ou janvier, au moment de la soutenance.

5.1.1 Mise à disposition du sujet

Une première version du sujet sera soumise début octobre pour vous permettre de commencer à travailler et pour susciter vos questions et demandes de précisions. Cette première version de cadrage est susceptible de nombreuses évolutions au cours du mois d'octobre, notamment en fonction des réponses à vos questions.

5.1.2 TP de lancement (TPP1)

Chaque groupe effectue un premier TP (de 4h) vers la mi-octobre. Ce sera l'occasion de constituer vos équipes, de prendre connaissance du sujet, effectuer vos premières recherches, initialiser votre travail d'élaboration.

Groupe A le samedi 27 octobre matin.

Groupe B le lundi 8 octobre après-midi.

Groupe C le samedi 10 novembre matin.

Groupe D le samedi 20 octobre matin.

Groupe PL le samedi 20 octobre matin.

5.1.3 Soutenance intermédiaire (S1)

Une soutenance intermédiaire (6h30 par groupe) aura lieu vers mi-novembre pour nous assurer que toutes les équipes ont commencé à travailler. Cette soutenance sera l'occasion d'un point individualisé (par équipe) sur la base d'un échange relativement informel. Il y aura cependant quelques points en jeu sur la note finale, qui seront notamment perdus par les équipes qui n'auraient pas commencé à travailler (sérieusement s'entend) :

Groupe A les samedis 17 et 24 novembre matin.

Groupe B le samedi 17 novembre matin, et le lundi 19 novembre matin.

Groupe C le vendredi 23 novembre après-midi et le samedi 24 novembre matin.

Groupe D les samedis 17 et 24 novembre matin.

Groupe PL les samedis 17 et 24 novembre matin.

5.1.4 TP d'appui intermédiaire (TPP2)

Un TP d'appui intermédiaire (4h par groupe) aura lieu dans la semaine, au pire dans la quinzaine qui suit la soutenance intermédiaire.

Groupe A le samedi 1er décembre matin.

Groupe B le vendredi 30 novembre après-midi.

Groupe C le samedi 8 décembre matin.

Groupe D le samedi 1er décembre matin.

Groupe PL le mardi 11 décembre après-midi.

5.1.5 Soutenance finale (S2)

Soutenance finale (6h30), avant les vacances de Noël.

Groupe A le lundi 17 décembre après-midi et le mardi 18 décembre matin.

Groupe B le mardi 18 décembre matin.

Groupe C le lundi 17 décembre après-midi et le mardi 18 décembre après-midi.

Groupe D le samedi 15 décembre matin et les lundi 17 et mardi 18 décembre après-midi.

Groupe PL le mardi 18 décembre après-midi.

Sauf pour le groupe D, vos tuteurs de projet sont vos animateurs de TP (A, C : Franck Lepoivre, B : Michel Landschoot, D : Itheri Yahiaoui, PL : Kaïs Klai).

5.2 Constitution des équipes

Binômes par groupe de TD. Trinômes interdits, sauf un unique pour groupe TD à nombre impair de personnes.

Les délégués de groupe sont chargés de produire une liste des équipes et des ordres de passage sur les plages horaires réservées pour les soutenances.

Il vous faudra avoir indiqué votre équipe à votre délégué avant le **vendredi 26 octobre au soir, au plus tard**. Au delà, votre équipe sera constituée d'office.

En cas de dysfonctionnement problématique au sein d'une équipe, n'attendez pas le dernier moment pour l'évoquer : on peut toujours trouver une solution.

5.3 IHM et technologies tierces parties

Vous pouvez utiliser les technologies de votre choix.

Les technologies tierces parties employées devront être référencées dans votre livrable, et notamment rappelées dans les annexes de votre rapport.

Il est interdit d'utiliser des librairies existantes pour vos SDD et algorithmes clés (en d'autres termes, pas de librairie, sauf librairie graphique).

Notez enfin que l'appréciation de l'IHM n'est qu'un constituant accessoire de la note. Évitez d'y investir trop de temps, au détriment de la recherche de la meilleure performance algorithmique. Chaque année, malgré cette mise en garde, de nombreux élèves sont frustrés par une piètre rétribution après un investissement significatif sur cette partie.

5.4 Évaluation

5.4.1 Comment gagner des points

Critères par ordre décroissant d'importance :
- Le programme compile.
- Le programme s'exécute sans planter.
- Le programme fait ce qu'on attend de lui (partiellement, complètement, avec des plus).
- La trace d'exécution est lisible / intelligible.
- Le code source est efficace (SDD, algorithmes efficaces).
- Le code source est propre (indentations, identifiants intelligibles, commentaires bien dosés).

5.4.2 Comment perdre des points

Critères par ordre décroissant de gravité :
- Même code, même partiel, livré par plusieurs équipes : 0 pour les équipes impliquées.
- Retard de livraison : 2 points par jour au *prorata temporis*.
- Non respect des consignes de packaging du livrable : jusqu'à –3 pts suivant les cumuls.

5.4.3 Travail minimum pour espérer avoir 10

Le périmètre fonctionnel minimal consiste à permettre de jouer aux 4 applications, sans *intelligence artificielle* (résolution automatique). Il faut donc permettre le chargement du jeu dans sa configuration initiale, permettre au joueur de déplacer les pièces en respectant les contraintes de déplacement, de détecter la fin de partie.

Ce périmètre fonctionnel minimum permet d'être noté sur 12 points : obtenir 12 serait alors la marque d'une réalisation parfaite sur ce périmètre (fonctionnement, ergonomie, etc). En d'autres termes, en l'absence de résolution automatique, il sera bien difficile d'obtenir la moyenne.

6 Précisions sur le livrable

6.1 Contenu du livrable

- Vos fichiers sources.
- SURTOUT PAS D'EXÉCUTABLE !
- Un script de compilation, déploiement et paramétrage pour Win32 accompagné d'un manuel d'installation minimal mais suffisant pour ne pas y passer plus de trois minutes.
- Vos fichiers de paramétrage et de configurations qui décrivent les différents jeux supportés par votre plate-forme (votre canevas).
- Votre documentation (Rapport et autres documents) au format PDF.
- Un fichier `README.txt` qui comprend impérativement deux sections :
  - Un état d'avancement honnête et fiable de vos travaux en 10 lignes maximum.
  - Les références aux sources qui ont pu inspirer votre conception, la reprise de code étant formellement interdite.
  - Et en option, une mise en avant de fonctionnalités supplémentaires que vous aurez choisi de réaliser dans le but d'obtenir une meilleure note.

6.2 Contenu du rapport

Un sommaire détaillé avec pages numérotées.

L'analyse fonctionnelle générale D'environ cinq pages, elle explique quelles sont les principales données traitées et quelles sont les principales fonctionnalités du programme, comment elles sont organisées dans le programme. Cette analyse aboutit à la présentation des modules fonctionnels qui sont décrits en détail dans l'analyse détaillée. L'analyse générale doit préciser le ou les choix importants que vous avez faits pour la conception de votre projet.

L'analyse fonctionnelle détaillée D'environ dix pages, elle précise, pour chaque module, son utilité, ses entrées et ses sorties, les données qu'il modifie. Il faut décrire les données utilisées et les algorithmes mis au point (logique de manipulation) avec rigueur, concision et clarté, le tout accompagné de schémas. C'est un travail de synthèse, de pédagogie et de communication ; les extraits de code sont proscrits de vos rapports.

Annexes Elles présentent au moins les technologies tierces employées. Et comme option recommandée, bibliographie et webographie listent les références qui ont permis d'étayer votre analyse et votre conception.

6.3 Livraison

Une seule remise des travaux par équipe ! ! (la double remise peut être sanctionnée dans la composante rigueur / discipline de la note).

Remise au plus tard le **14 décembre 2012**. Dès le lendemain (le 15), application d'une pénalité de 2 points par jour (au *prorata temporis*).

Le livrable se présente sous la forme d'une archive RAR de nom :

`PROJET SDD L2 GP - NOM 1 - NOM 2.RAR`

Envoi par email à l'adresse de votre tuteur de projet, ou par tout autre moyen de substitution qu'il vous indiquerait.

Objet `PROJET SDD L2 GP - NOM 1 - NOM 2`

Franck Lepoivre (groupes A et C) : `franck.lepoivre@gmail.com`,

Michel Landschoot (groupe B) : mlandsnet@yahoo.fr,
Itheri Yahiaoui (groupe D) : itheri.yahiaoui@inria.fr,
Kaïs Klai (groupe PL) : kais.klai@lipn.univ-paris13.fr

7 Annexe 1. Références

7.1 Bibliographie

7.1.1 Théorie combinatoire des jeux

La *bible* de la théorie combinatoire des jeux publiée en 1975 par Conway (l'inventeur du monumental *jeu de la vie*) et ses acolytes, Elwyn Berlekamp et Richard Guy.

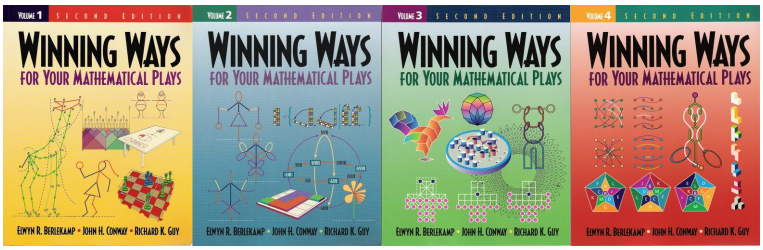

FIGURE 10 – Tétrominos de Tétris

La partie qui nous intéresse commence (dans le volume 4) à la page 877 : les jeux de type Taquin ou Âne Rouge y sont traités (modélisation et stratégies de résolution).

7.2 Webographie

7.2.1 Premières lectures

Puzzles à pièces coulissantes :

- `http://fr.wikipedia.org/wiki/Casse-tête_de_déplacements`
- `http://en.wikipedia.org/wiki/Sliding_block_puzzle`

Le Taquin :

- `http://fr.wikipedia.org/wiki/Taquin`
- `http://en.wikipedia.org/wiki/Fifteen_puzzle`

L'Âne Rouge :

- `http://fr.wikipedia.org/wiki/L'Âne_rouge`
- `http://en.wikipedia.org/wiki/Klotski`

Dad's Puzzler :

- `http://www.cs.brandeis.edu//~storer/JimPuzzles/ZPAGES/zzzDadsFamilySet.html`

Century de Conway – `http://fr.wikipedia.org/wiki/Jeu_century`

7.2.2 Très bon article de synthèse, au cœur du sujet

`https://www.maa.org/editorial/mathgames/mathgames_12_13_04.html`

7.2.3 Études algorithmiques connexes

Défi algorithmique de Developper.com : recherche de solution de l'Âne Rouge `http://c.developpez.com/defis/3-Ane-Rouge`

7.2.4 Répertoires de puzzles à pièces coulissantes

Nick Baxter Sliding Block Puzzles `http://www.puzzleworld.org/slidingblockpuzzles`
`http://www.johnrausch.com/SlidingBlockPuzzles/modern.htm`
`http://www.cs.brandeis.edu//~storer/JimPuzzles/ZPAGES/zzzDadsPuzzler.html`

8 Annexe 2. Histoire des puzzles à pièces coulissantes

Block puzzle, Path puzzle, Kwirk, Professor Layton

Century, SuperCompo, Quzzle

Famille des 4x5 : Dad's Puzzler, Nine Block, Quzzle / Quzzle Killer, Traffic Jam, Red Donkey, Century, Super Century, Century And A Half, Little House, Ushi, Hole in One, King Out, etc. Hordern's book

9 Annexe 3. Répertoire de puzzles à pièces coulissantes

10 Annexe 4. Format de base pour la représentation des états et des enchaînements

10.1 Représentation des états et enchaînements d'états de jeu

Ce format de représentation est une convention simple qui permet de nous entendre sur le format des fichiers de configuration de vos jeux. Il permettra notamment de vous mettre en compétition, de tester vos réalisations sur la base des mêmes fichiers.

C'est le format le plus répandu dans la littérature en ligne : on représente un état directement comme il est *vu*, c'est à dire en reproduisant la géométrie du plateau, dont la grille, et en indiquant pour chaque place, par un code numérique, l'état d'occupation de cette place [5] :

X représente une place qui ne peut être occupée.

. représente une place inoccupée (c'est une place vide).

une lettre autre que . ou X représente une place occupée par la pièce dont la lettre est l'identifiant.

La série d'états d'occupation est donnée sous forme d'une matrice rectangulaire des codes ci-dessus, donnée ligne par ligne, où les codes successifs d'une même ligne sont juxtaposé sans espace blanc.

Pour décrire une séquence d'états dans un fichier, on se contente de décrire les états les uns après les autres, en utilisant un retour à la ligne comme séparateur entre deux états successifs.

Attention : cette spécification n'est en aucun cas une prescription du format des structures de données en collaboration avec lesquelles devront travailler vos algorithmes [6].

10.1.1 Exemple simple avec l'Âne Rouge

On ne donne que les deux premiers états, et le dernier (mais il y a bien 80 états en tout) :

Listing 1 – Résolution de l'Âne Rouge

```
FJJG
FJJG
HEEI
HABI
C..D

FJJG
FJJG
HEEI
HABI
C.D.

...

FHGI
FHGI
ABEE
```

5. Pour permettre de représenter des formes de plateau non rectangulaires, on s'autorise la représentation de places qui ne peuvent être occupées par une pièce à l'aide la lettre *X*.

6. En effet, ce serait assurément une approche peu performante puisqu'il n'y a aucune recherche d'optimisation de l'empreinte mémoire de chaque état, ni aucune recherche de la forme la plus adaptée aux opérations à effectuer pour passer d'un état au suivant.

```
.JJC
.JJD
```

10.1.2 Exemple plus complexe avec le Junk's Hanoi by Junk Kato

Comment représenter, dans un fichier, l'enchaînement des 160 mouvements qui permettent de résoudre le Junk's Hanoi dont l'animation est visible ici : `http://www.youtube.com/watch?v=1sJUML49XVA`

Listing 2 – Résolution du Junk's Hanoï

```
AAAXXXXXXX...
BBBBXXXXX....
CCCCCXXX.....
DDDDDDX......
EEEEEEE......
XXXXXX.......
XXXXXXX......
XXXXXXXX.....
XXXXXXXXX....
XXXXXXXXXX...

AAAXXXXXXX...
BBBBXXXXX....
CCCCCXXX.....
DDDDDDX......
............
XXXXXXEEEEEEE
XXXXXXX......
XXXXXXXX.....
XXXXXXXXX....
XXXXXXXXXX...

AAAXXXXXXX...
BBBBXXXXX....
CCCCCXXX.....
......XDDDDDD
............
XXXXXXEEEEEEE
XXXXXXX......
XXXXXXXX.....
XXXXXXXXX....
XXXXXXXXXX...

...

...XXXXXXX...
....XXXXX....
.....XXX.....
......X......
............
XXXXXXEEEEEEE
XXXXXXXDDDDDD
```

```
XXXXXXXXCCCCC
XXXXXXXXXBBBB
XXXXXXXXXXAAA
```

10.2 Fichiers de configuration de jeu

La description précédente permet de représenter des états initiaux de jeu et des séquences d'états qui conduisent à une résolution.

Elle ne permet en revanche pas de décrire une configuration de jeu comme la simple donnée d'un état de départ et d'un état but. En particulier, pour représenter un couple état initial / état but, il faut pouvoir représenter l'*abstraction d'un état but*, c'est-à-dire un ensemble d'états qui présentent une même caractéristique.

Soyons plus précis : par exemple, pour le Taquin, l'état but est unique, et la configuration qui recompose une une combinaison particulière des pièces, une image singulière. Mais par exemple, pour l'Âne Rouge, le but est atteint en tout état qui positionne la pièce principale 2×2, l'Âne Rouge, en cette place qui permet de l'extraire du plateau de jeu. Dans ce cas, plusieurs états sont satisfaisants, notamment, tous les états qui positionnent ainsi la pièce 2×2, abstraction faite des autres pièces.

Pour représenter cet état abstrait, ou l'on fait abstraction de certaines pièces, on utilisera le symbole # pour représenter un contenu quelconque d'une place. Par exemple :

Listing 3 – Etat but de l'Âne Rouge

```
####
####
####
#JJ#
#JJ#
```

Représente l'abstraction de l'état but de l'Âne Rouge.