Présentation

(Overview)

Cube Explorer n'est pas comme un des nombreux programmes qui se contentes de simuler le Cube, ou de résoudre des Cubes brouillés par de longues tractations. Cube Explorer met en œuvre un algorithme sophistiqué très puissant (l' Algorithme en deux phases).
Cet algorithme donne généralement une résolution de la séquence de 19 coups en moyenne dans les secondes qui suivent, seulement un ou deux coups de plus que la solution parfaite. Cube Explorer met également en œuvre un algorithme pour trouver la solution idéale: une solution qu'il est probablement impossible à surpasser. Avec un équipement PC de l'année 2005, il prend généralement moins d'une heure pour trouver la solution.

Cube Explorer vous aide également à trouver de jolis motifs (il faut exécuter très peu de manœuvres pour les générer).
Cube Explorer cherche dans l'univers des cubes productibles la liste de tous les cubes correspondants à votre modèle.

S'il vous plaît notez: Cube Explorer n'est pas dérivé de, n'est pas associé avec et n'est pas approuvé ou commandité par le propriétaire de la marque du Rubik's Cube . C'est Seven Towns Limited, le fabricant et distributeur mondial du puzzle en trois dimensions Rubik's Cube et fournisseur d'une version électronique du casse-tête par l'intermédiaire de son site web officiel à **http://www.rubiks.com/**.
Cube Explorer est un produit éducatif et non-commercial. Il est gratuit et le résultat de la recherche scientifique.

Définitions De Base

(Basic Definitions)

Le cube se compose de 8 petits cubes de coin (cubies), de 12 petits cubes de bord (cubies)et de 6 petits cubes centraux(facelets).
Les bords peuvent être renversés,et les coins peuvent être permutés. Cela signifie l'orientation de ces petits cubes peut changer dans l'espace.
Les 6 faces différentes s'appellent U(p) H(aut),D(own) B(as),R(ight) D(roite), L(eft) G(auche), F(ront) A(vant)et B(ack) A(rriere).
Tandis qu'U signifie faire tourner la face supérieure de 90 degrés dans le sens des aiguilles d'une montre, U2 dénote la faire tourner de 180 degrés et U' dénote un quart de tour soit 90 degrés dans le sens contraire des aiguilles d'une montre. Un ordre de mouvement du cube comme U D R 'D2 s'appelle une manœuvre
Le nombre de mouvements s'appelle la longueur de manœuvre.
Une manœuvre peut être un solutionneur ou un générateur. Tandis qu'un solutionneur est une manœuvre qui reconstitue l'état original du cube quand elle est appliquée à un cube brouillé, un générateur est exactement l'inverse d'un solutionneur,elle est appliquée à un cube rangé en ordre avec pour résultat un cube brouillé.
Cela se consoit si vous voulez jouer à créer de jolis modèles.

Algorithmes de résolution du Cube

(Cube Solving Algorithms)

La stratégie la plus simple pour résoudre le cube consiste à procéder par étapes, chaque étape détermine l'emplacement ou l'orientation d'un certain nombre de coins ou de bords. Lorsque vous vous déplacez dans le sens prescrit par l'algorithme, vous peut voir chacune des faces de prendre une couleur unique. Même si elle est appropriée pour restaurer le Cube à la main, elle n'est pas optimale en ce qui concerne la longueur totale des manœuvres, car elle ne tient pas compte du groupe de mathématiques structurant le Cube. Ce type d'algorithme nécessite habituellement plus de 70 coups.

Le premier qui a développé un algorithme qui prend en compte la structure du groupe a été le Cube de Morwen Thistlethwaite mathématicien en 1980. Il a travaillé par le biais d'une série de quatre sous-groupes du cube.
Les informations sur la façon de faire sont consignées dans quatre tables. Un ordinateur est devenu l'outil pour les utiliser. Restaurer le Cube a nécessité 52 coups tout au plus.

Hans Kloosterman a amélioré cet algorithme qui ne nécessita pas plus de 42 coups en 1990 grâce à une organisation différente des tableaux.

J'ai développé l'algorithme en deux phases, en 1992, pour une machine avec 1 Mo de mémoire RAM. Une limite maximale du nombre exact de manœuvres ne peut être trouvée parce que chaque cas est traité individuellement et qu'il est impossible
d'examiner tous les cas. Mais en moyenne, vous pouvez parvenir à une solution de moins de 19 déplacements dans un délai assez bref.

Richard E. Korf à appliqué des idées générales sur la théorie des algorithmes heuristiques de recherche en intelligence artificielle au cas particulier du cube en Mai 1997 et a trouvé une courte manœuvre pour 10 séquences générées aléatoirement sur des cubes. Le programme tournait sur un Sun Ultra Sparc Model-1 workstation et avait besoin d'environ 82MB de RAM. Pour trouver chacune de ces solutions un délai moyen de 5 jours était exigé. Mais certains cas ont nécessité des mois ou des années de trouver une solution. L'algorithme en une passe utilisé s'est avéré très similaire à celui utilisé à chaque passe de l'algorithme en deux phases.

En Juillet 1997, en profitant de la symétrie du cube, Michael Reid, a été en mesure de stocker efficacement plus d'information.
La base de données utilisée contenait en mémoire toute la première phase de l'algorithme en deux passes (Ndt:le fameux Tow-Phase-Algorithm) , elle était assez rapide pour tourner comme un solutionneur une phase. Il a fait tourner son programme sur une machine avec 128 Mo de RAM equipée d'un Processeur Pentium Pro 200 MHz . Le programme a trouvé une série de manœuvres plus courte et vingt fois plus rapide que le programme Korf .

En 2001, j'ai adopté mon algorithme à deux phases aux capacités améliorées des PC (avec environ 128 Mo de RAM).
la Phase 1 est aussi chargée en mémoire , de sorte que c'est est extrêmement rapide. La partie de Optimal du solutionneur utilise la phase 1 du solutionneur en 1 phase avec les manœuvres les plus courtes pour trouver des séquences de la même façon que dans la mise en œuvre Michael Reid.

La limite pour un système d'exploitation 32 bits comme Windows XP étant de 4 Go de RAM, j'ai adopté la solution optimale pour ce contenance de mémoire. Elle tourne environ 15 fois plus vite que la phase 1 du solutionneur optimal 2001. Utilisez la 's'-version de Cube Explorer et lancez le large Optimal Solutionneur du Menu Option pour utiliser cette fonctionnalité.
En utilisant les quatre core d'un processeur Intel Core i7 (4 processeurs intégrés) sur un ordinateur Windows Vista 64 bits, l'optimal solutionneur a résolu de façon optimale environ 290 cubes aléatoire dans l'heure (2009).

L'algorithme à deux phases (en bref)

(The Two-Phase Algorithm (in brief))

La description qui suit est destiné à vous donner une idée de la façon dont l'algorithme fonctionne. elle est tout à fait technique et peut être difficile à comprendre. Cette information n'est pas nécessaire pour utiliser le programme, et vous n'avez pas besoin d'en être conscient pour utiliser le programme.

Si vous tournez les faces d'un cube résolu et n' utilisez pas les mouvements R, R, L, L, F, F ', B et B', vous ne générerez qu'un sous-ensemble de tous les cubes. Ce sous-ensemble est désigné par G1 = <U,D,R2,L2,F2,B2>. Dans ce sous-ensemble, l' orientations des coins et des bords ne peut être changé. C'est ainsi, l'orientation d'un bord ou un coin à un endroit est toujours la même. Et les quatre bords de la tranche UD (entre la face-U et la face-D), restent isolés dans cette tranche.

Au cours de la phase 1, l'algorithme cherche les manœuvres qui vont transformer un cube à brouillés en G1. Ainsi, l'orientation des coins et des bords ne doit pas changer et les bords de l'UD-tranche doivent être transférées dans cette tranche.
Dans cet espace abstrait, un mouvement transforme juste un triple (x, y, z) en un autre triple (x ', y', z '). Tous les cubes de G1 ont le même triple (x0, y0, z0), et c'est l'objectif de la phase 1.

Pour connaître l'état de cet objectif le programme utilise un algorithme de recherche qui, en termes de la recherche actuelle, est appelé Un approfondissement itératif de A* avec une fonction heuristique limite basse (IDA *). Dans le cas du Cube, cela signifie qu'il parcourt toutes les manœuvres jusqu'aux plus longues. La fonction heuristique h1 (x, y, z) estime le nombre de coups qui sont nécessaires pour solutionner chaque état du cube (x, y, z). Il est essentiel que les fonction ne surestime pas ce nombre. Depuis Cube Explorer 2, il donne le nombre exact de mouvements qui sont nécessaires pour atteindre l'objectif de l'État dans la phase 1. L'heuristique permet d'élaguer la génération des manœuvres, ce qui est essentiel si vous ne voulez pas attendre très, très longtemps avant que l'objectif de l'État soit atteint.
La fonction heuristique h1 est une table verrouillée basée en mémoire et permet de prévoir jusqu'à 12 coups à l'avance.

Dans la phase 2 de l'algorithme restitue le cube dans le sous-groupe G1, en utilisant seulement les déplacements de ce sous-groupe. il restaure la permutation des 8 coins, la permutation des 8 bords de la face-U et de la face-D, la permutation des 4 bords-UD de la tranche.
La fonction heuristique h2 (a, b, c) estime que le nombre de coups qui sont nécessaires pour atteindre l'état de l'objectif , car il y a trop d'éléments différents dans G1.

L'algorithme ne s'arrête pas lorsque la première solution est trouvée, mais continue à rechercher des solutions plus courtes en exécutant dans la phase 2 des sous-solutions optimales de la phase 1. Par exemple, si la première solution est de 10 coups de la phase 1 suivi de 12 coups en phase 2, la deuxième solution pourrait avoir 11 coups de la phase 1 dont seulement 5 se déplace dans la phase 2. La longueur de manœuvres de la phase 1 augmente la longueur de manœuvres de la phase 2 diminue.
Si la longueur de la phase 2 atteint zéro atteint zéro, la solution est optimale et l'algorithme s'arrête.

Dans le contexte actuel le but de Algorithm-deux phases n'est pas de chercher des solutions d'ensembles optimaux mais d'utiliser ceux-ci en fond pour la phase 2. Ceci augmente considérablement la vitesse de résolution.
Donc,utilisez le Solutionneur Optimal seulement si vous voulez trouver une manœuvre optimale.

Explorer le Cube

(Exploring the Cube)
Cube Explorer se compose de plusieurs feuilles à onglet et une fenêtre principale:
Il y a des onglets pour les fiches suivantes

1. L'éditeur de facettes(**The Facelet Editor**), qui vous permet de modifier les couleurs d'un cube, les mouvements à appliquer au cube ou à le résoudre.

2. L'éditeur de figures,(**Pattern Editor**) qui vous aide à créer Jolis motifs.

3. l'Editeur de symétrie (Symmetry Editor) , qui vous aide à trouver des symétries ou des Anti-Symmetries à un cube.

4. La Web Cam Tabbed Sheet où vous pouvez utiliser une webcam pour entrer dans les faces d'un cube.
La sortie est affichée dans la fenêtre principale (Main Window),

Le but principal de Cube Explorer est de trouver des générateurs et des solutionneurs ou des modèles calculés aussi court que possible.
À partir d'un cube dans sa position résolue, un générateur est une manœuvre qui va le transformer en position affichée dans la fenêtre principale. Inversement, à partir d'un cube dans la position affichée dans la Fenêtre principale, un solveur est une manœuvre qui va la transformer en position résolue.

Installation de Cube Explorer

(Installing Cube Explorer)

Cube Explorer à uniquement besoins des fichiers cube4xx.exe , de cube.chm et éventuellement, de l'applet java pour AnimCube.jar
pour fonctionner. Le fichier cube.ini contient une liste de paramètres par défaut. Vous devez copier tous ces fichiers dans le même dossier. A la première mise en route, certains tableaux sont calculés et stockés dans votre dossier Cube Explorer. Selon votre matériel, cela peut prendre jusqu'à 30 minutes. A la prochaine exécution de Cube Explorer, les tables sont directement chargés à partir de votre disque dur dans la mémoire, ce qui est considérablement plus rapide. Les tableaux pour les fonctions de base besoin d'environ 70 Mo de mémoire. Cube Explorer a un très puissant algorithme implémenté pour résoudre les Cube, et ces tableaux sont une partie essentielle de l'algorithme.

Cube Explorer ne modifiera pas ou ne créera de fichiers à l'extérieur du dossier, donc, si vous décidez de désinstaller Cube Explorer il vous suffit de supprimer ce dossier.

L'éditeur de facette

(The Facelet Editor)

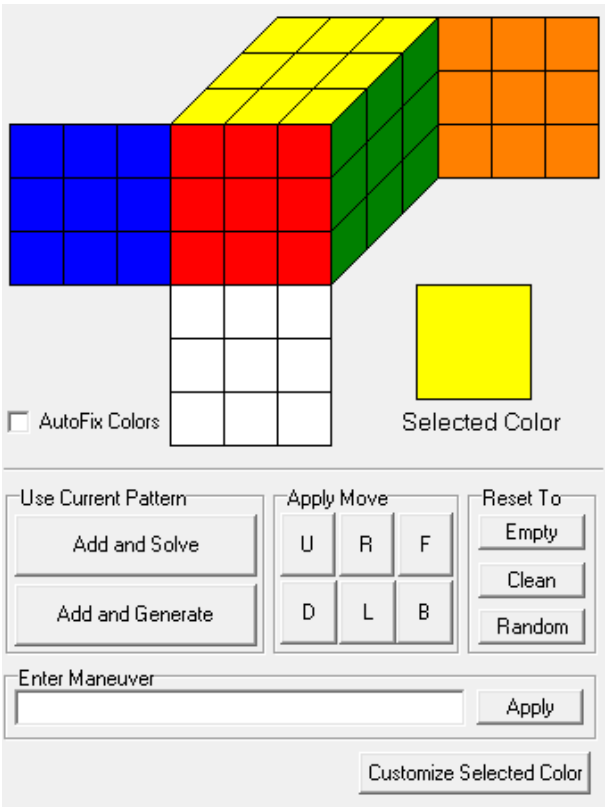
La Modification de facettes est très simple,
mais il y a quelques caractéristiques qui ne sont pas évidente, voici donc un résumé:

•
Clique gauche au centre d'une facette pour sélectionner une couleur.

•
Clique gauche toute autre facette pour la remplir avec la couleur sélectionnée. Shift-Clique gauche et Ctrl-Clique gauche ouvre des options additionnelles
pour résoudre des cubes incomplets .

•
Clique-droit au centre pour modifier les rotations des centres, si vous avez permis l'utilisation de rotation des facettes centrales
(**Use Center Facelets Twists)**
dans l'option Dialogue.

•
Clique-droit toute autre facette pour effacer la couleur.
Lorsque vous entrez un cube que vous voulez résoudre ou de générer, il peut être utile d'utiliser le bouton remet en ordre (**Clean)** en premier.



Avec ajouter et résoudre(**Add and Solve**), le cube est transféré dans la fenêtre principale
et le deux phases-Algorithme le résout. Si vous utilisez Ajouter et Générer (**Add and Generate),** le générateur
calcule un motif. Vous pouvez permuter le solveur et le générateur dans la Fenêtre principal
en cliquant sur le bouton fournit à cet effet.

Le bouton Hasard (**Random**) est utile si vous voulez vous convaincre de la performance de l'algorithme.
L'un des 43.252.003.274.489.856.000 cubes possible est choisi au hasard avec la même probabilité.

Avant d'ajouter le cube de la fenêtre principale, une vérification se fait pour savoir si un cube isomorphes est déjà présent dans la fenêtre principale.

dans ce cas vous recevez un message.

Vous pouvez saisir ou coller et déplacer dans la boîte de manoeuvres (Enter Maneuver).En plus des mouvement standards,les mouvements de tranches et tranches inversées (e.g Ra, nous'...)sont également acceptés.

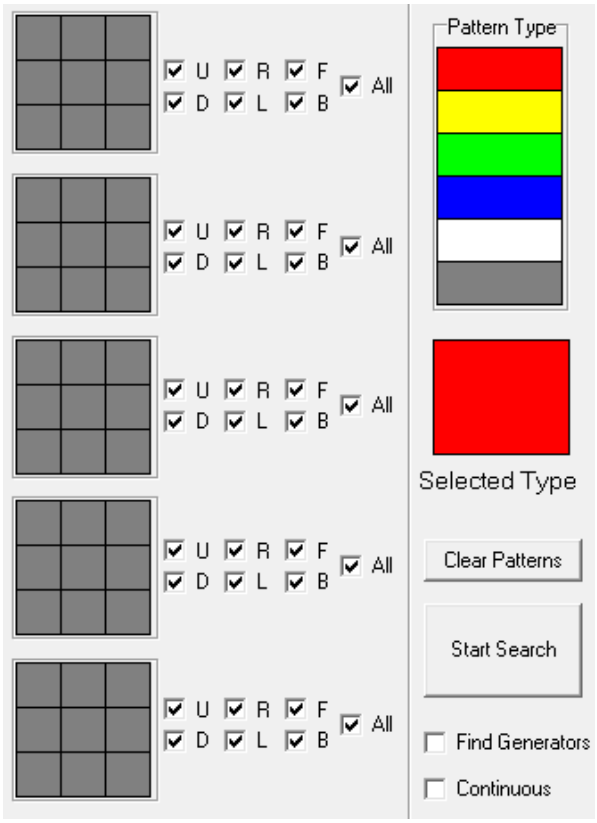
Si les couleurs de votre cube sont différentes, vous devez sélectionner une couleur en cliquant sur l'un des six cubes centraux.

Ensuite,
utilisez le bouton Personnaliser la couleur sélectionnée (**Customize selected Color)**.

Si couleurs automatique
(**Autofix Colors**)est cochée, une correction
automatique d'erreur se fait quand vous fixez les couleurs des faces. Même si cela est utile à bien des situations, cela peut être très ennuyeux dans certains cas.

L'éditeur de figures

(The Pattern Editor)



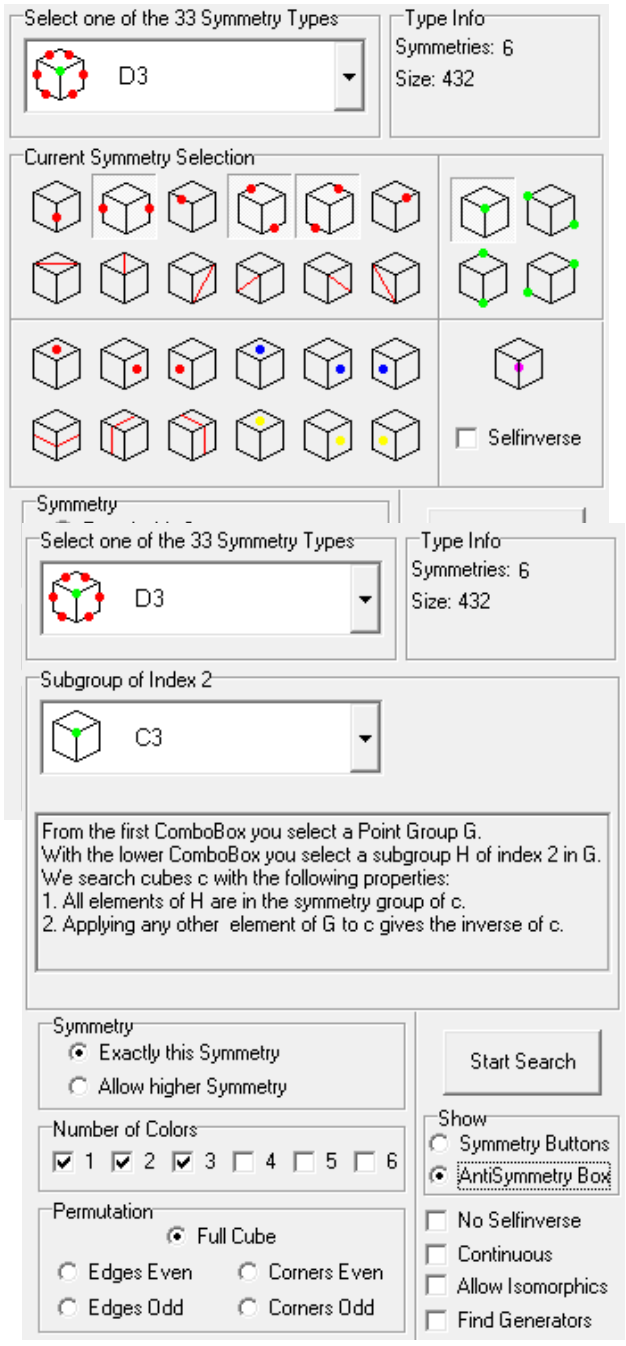
Vous utiliserez le l'éditeur de figures si vous voulez
créer de jolies figures sur un cube.

Il est important de comprendre que les couleurs vous avez choisi en cliquant sur Type de figure
(**Pattern Type)**
ne représente pas les les vraies couleurs du cube, elles servent seulement de définir une structure.
Dans l'exemple de gauche, vous recherchez des cubes, dont le coté droit, gauche, face avant ou arrière font un carré 2 x 2 (1. pattern modèle), et une face unie (2. Pattern modèle) ou un damier (3 pattern modèle) sur le haut ou le bas.
Si **toutes** faces sont grises (comme dans les exemples de modèles 4. et 5.),la définition est ignorée.
Si vous cochez la case Rechercher les générateurs **(Find Generator**), le TwoPhase - Algorithme sera utilisé pour les cubes qui correspondent à la structure des modèles.
s'il existe plusieurs solutions de cubes isomorphes,
seulement l'un d'eux sera ajouté dans la fenêtre principale.

Un motif est généré en **continu** si deux faces sont voisines le long d'un bord. C'est-à-dire que si deux arrêtes ou deux coins voisins ont la même couleur
sur la première face, si et seulement s'ils ont la même couleur sur la deuxième face. ce genre de motif Continu est très prisé pour réaliser des motifs d'une beauté exceptionnelle.

Editeur de symétrie

(The Symmetry Editor)



Avec l'Editeur de symétrie vous rechercherez la symétrique et l'antisymétrique des cubes. Ce sont des cubes dont certaines opérations ne peuvent varier compte tenu de la symétrie.

Il y a 48 rotations géométriques, des réflexions etc, qui cartographies un cube sur un cube. les sous sélections de ces cartes de définir les sous-groupes de symétrie du cube. Il existe 33 différents sous-groupes essentiels.
Vous pouvez définir ces sous-groupes en appuyant sur les boutons ou par sélection à partir de la combobox.
Vous pouvez également analyser les symétries et antisymétries d'un cube dans la Fenêtre
principale en cliquant dessus. Un bouton de symétrie est enfoncé, si le cube correspond à cette symétrie et ne l'est pas si le cube correspondant à une antisymetrie.

Avec démarrer la recherche (**Start Search**),tous les cubes qui respectent ce jeu de symétries sont calculés.

Si vous utilisez une haute symétrie(**Allow higher Symmetry**) , les cubes trouvées sont autorisés à avoir d'autres symétries données sans les boutons pressés.
Vous pouvez définir combien de couleurs pour une face en cochant un nombre de Couleurs(**Number of Colors**).

Si vous voulez analysez uniquement les permutations d'arêtes ou de coins vous pouvez utiliser le bouton radio dans la zone du groupe **Permutation.**
Vous devez décider si vous souhaitez rechercher des de permutations paires ou impaires .

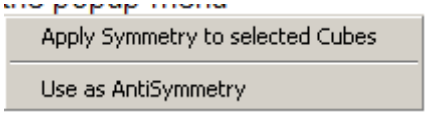
Pas d'auto inversion (**No Selfinverse**) exclut les cubes qui ont une auto inversion en sortie vers la Fenêtre Principale **(Main Windows)**.

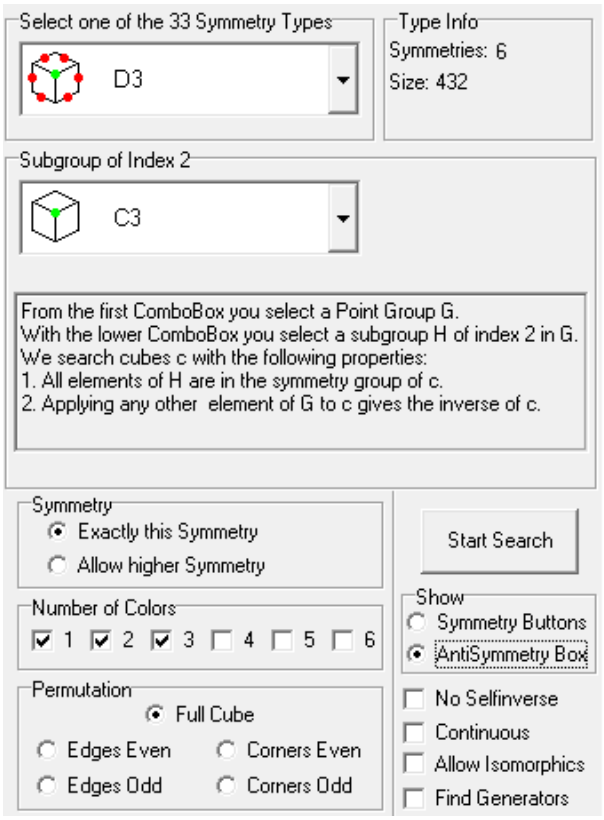
L'explication de Continuité (**Continous**)et de Recherche de générateurs (**Find Generators**) est donné dans la page d'aide de l'éditeur de figures.

Si vous cochez Autoriser Isomorphiques (**Allow Isomorphics**), les cubes

isomorphiques sont traités différements.

Vous pouvez explorer antisymmetrie avec un clique droit sur le bouton symétrie (symmetry)et en sélectionnant se servir de l'anti-symétrie (Use as AntiSymmetry) du menu popup qui apparait ou en cochant le bouton radio boite anti-symétrie (AntiSymmetryBox) -voir ci-dessous.





Nous demandons une antisymétrie pour un cube en symétrie S, si S transforme le cube en inverse. L'application de S avec l'opérateur d'inversion T redonne le cube original à nouveau.
Pour chaque type de symétrie dans boite combo supérieure (**upper combobox**) il y a habituellement plusieurs sous-groupes d'indice 2 dans la boite combo inférieure (**lower combobox**) qui définissent ensembles certains types d'anti-symétrie.

La mise en Tableau par Webcam

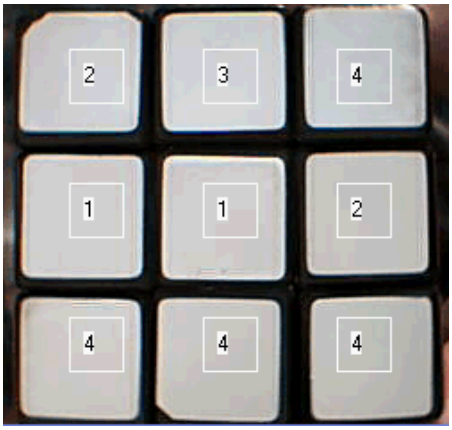
(The Webcam Tabbed Sheet)

Au lieu d'utiliser le Facelet Editor, il est possible d'utiliser une webcam pour saisir les faces d'un cube. Cela se fait très rapidement mais quelques points importants qui doivent être pris en considération si vous voulez scanner les faces avec succès.

1. Configuration de la Webcam

Sélectionnez la Webcam comme pilote de capture(**Capture Device Configuration**). Assurez-vous de brancher la webcam avant de lancer Cube Explorer, sinon son pilote ne s'affichera pas dans la liste des périphériques de capture (capture devices).

Appuyez sur le bouton configuration du périphérique de capture (**Capture Device Configuration**) pour ajuster la balance des blancs, l'exposition, le gain et la saturation: Mettez soit la face blanche d'un cube résolu soit une feuille de papier blanc face à la webcam. Activer la balance des blancs automatique, si s'est possible, et laisser la correction agir.



Sur l'image de gauche, la balance des blancs est OK. Les valeurs affichées pour la saturation(sat) sont faibles,elles doivent généralement être au-delà de 20.

Sur l'image de droite vous voyez une mauvaise configuration de la balance des blancs.

Ensuite, désactivez (**disable**) la balance automatique des blancs. Si vous ne désactivez pas la balance automatique des blancs, les couleurs ne seront pas reconnu correctement, **n'oubliez pas cette étape!**

Réglez l'exposition ou de gain de façon à ce que les couleurs ne semblent pas trop brillantes.

La couleur jaune est trop lumineuse sur l'image de gauche.

L'image de droite est OK. L'affichage des numéros de teintes, doit être entre 20 et 60 pour le jaune (voir

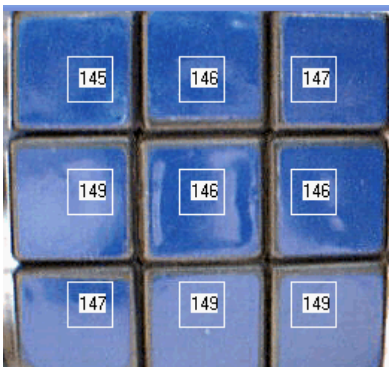
ci-dessous).

Si vous recevez le message
"Diminuer l'exposition ou de gain pour un résultat optimal! dans le coin supérieur gauche de la fenêtre de la Webcam c'est probablement un cas de luminosité. A nouveau, **désactivez** les paramètres d'exposition et de gain automatique ou vous courez vers les problèmes.

Il peut-être utile d'augmenter la saturation, mais en général cette étape n'est pas nécessaire.

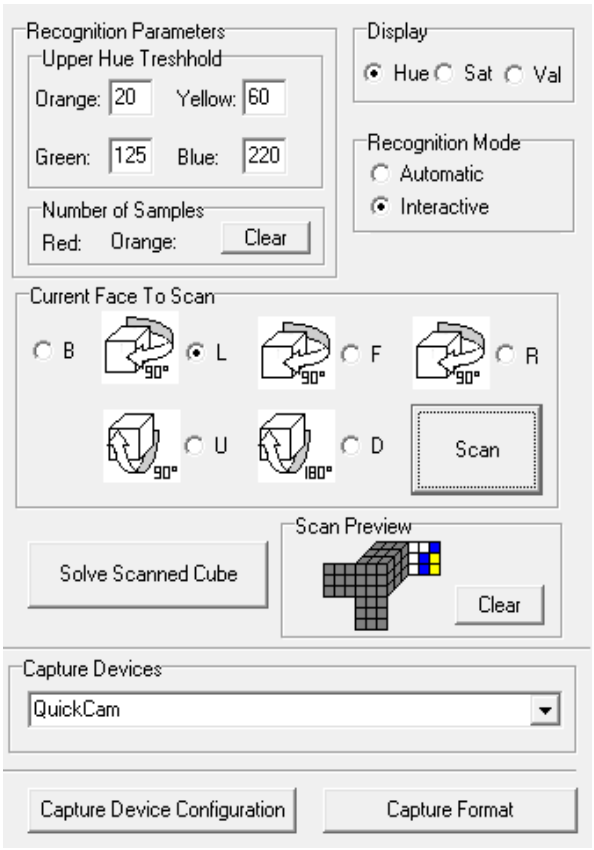
2. Mettre en place les paramètres de reconnaissance
Mettez le mode de reconnaissance automatique (reconition mode)et de vérifier le bouton radio Teinte est activé(dans Display Radio button **Hue**).
visionnez les différentes couleurs avec la webcam.

Les valeurs pour les teintes des faces orange et rouge doivent être inférieure à 20 ou plus de 220,
pour la face jaune moins de 60, pour la face verte moins de 125 et de moins de 220 pour la face bleu.
Si ce n'est pas le cas, vous devez changer de le Seuil Haut de la teinte (**Upper Hue Treshhold**)des couleurs, mais en général les valeurs par défaut sont valables pour les couleurs du cube.



Les teintes de bleu sont OK dans cet exemple.

Je vais vous expliquer plus tard comment distinguer le rouge et l'orange parce que cela dépend du mode de reconnaissance que vous choisissez automatique ou interactif.



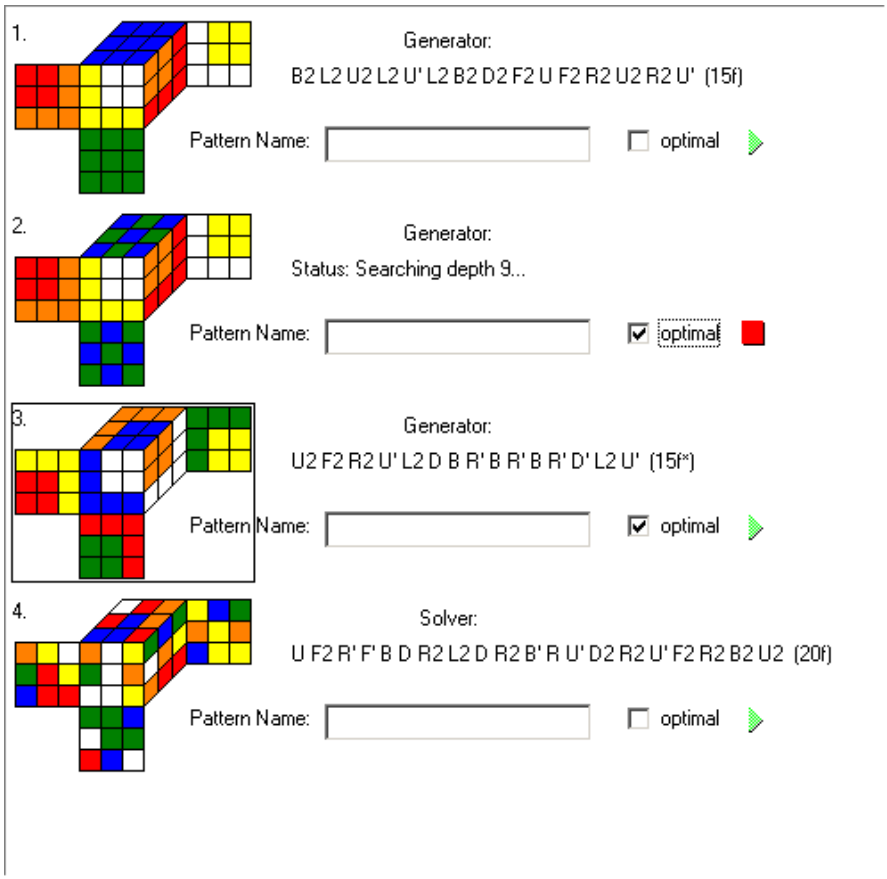
Si vous choisissez **le mode de reconnaissance Interactif (Interactive Recognition Mode)** (c'est le mode qui a le meilleur renvoie au cours de la procédure de numérisation),les couleurs sont identifiés en une fois à l'exception des faces rouge et orange où vous pouvez voir certains points d'interrogation (??).
Si vous cliquez sur points avec le bouton droit de souris, vous pouvez choisir la couleur de cet échantillon.
Les points d'interrogation ne disparaîtront pas tant que vous avez n'aurez pas fait 2 échantillons pour le rouge ou l'orange. Vous pouvez faire plusieurs échantillonnages (jusqu'à 9), si par exemple une face orange présente toujours comme rouge sous bien que modifiée.

Si vous choisissez le Mode de reconnaissance automatique (**Automatic Recognition Mode)**l'analyse des couleurs de la face ne se fait pas tant que vous que vous n'appuyez pas sur le bouton résoudre le cube scanné (**Solve Scanned Cube bouton**).
La distinction entre les faces rouge et orange est basée sur la couleur ou la valeur de ces couleurs. Si les teintes de rouge et orange, sont trop proches (ce que vous pouvez vérifier en réglant l'affichage sur teinte) vous pouvez régler la valeur dans le bouton radio.

Si vous **scannez** le cube, la face arrière est scanné en premier lieu, vous devez donc placer le cube comme le montre la photo de gauche, vous devez voir les faces U-F tandis que la caméra voie la face B avant de presser le bouton **Scan.**
Pour scanner la face L, tournez le cube comme montré dans les Boutons Radio B et L de la fenêtre "face courante à scanner" (**"courant Face To Scan" groupbox**) et appuyez sur le bouton scan.
Continuez jusqu'à ce que toutes les faces soient scannées.
Dans le mode interactif de reconnaissance que vous avez un bon rendu dans la fenêtre d'aperçu de numérisation (**Scan preview**)si vous analysez le cube correctement.
Cela fait appuyez sur le bouton Résoudre le Cube Scanné (**Solve scanned Cube**) . Si toutes les faces numérisées sont correctes,la résolution manœuvre est calculée.

La fenêtre principale

(The Main Window)



Dans la fenêtre principale, il existe plusieurs manières de procéder aux manipulations.

Appuyez sur ▶ si vous voulez lancer ou continuer l'algorithme de recherche de manœuvres.

Appuyez sur ■ si vous voulez mettre un terme au calcul en cours. Cube est un explorateur multitâches, il permet d'exécuter plusieurs calculs en même temps. Avant de fermer Cube Explorer, vous devez arrêter tous les calculs en cours d'exécution .

Si vous cochez la case **optimal**, la séquence de manœuvre la plus courte est calculée.
Cela peut prendre un certain temps et la profondeur de la recherche actuelle est affichée (cube 2 dans l'exemple ci-dessus).

•

Si le Solutionneur Optimal (**Optimal Solver**) a trouvé une solution,
la durée de la manœuvre est marquée d'un astérisque (cube 3 * 15 dans l'exemple ci-dessus)

•

En cliquant sur **générateur ou Solutionneur**(**Generator or Solver**), vous pouvez permuter ces deux manœuvres.

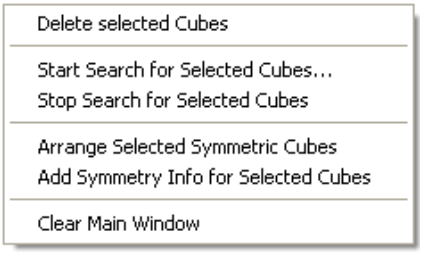
•

Pour supprimer un cube dans la fenêtre principale, sélectionnez-le avec le bouton gauche de la souris, puis appuyez sur la touche Supprimer, sélectionnez **"Edit**, puis Supprimer les Cubes sélectionnés (**Delete selected Cubes**")" dans le menu (ou le menu Popup, si vous cliquez avec le bouton droit dans la fenêtre principale, mais pas sur un cube). Si vous souhaitez supprimer plusieurs cubes à la fois, sélectionnez-les tous avant. Quand vous sélectionnez un cube avec le bouton gauche de la souris,

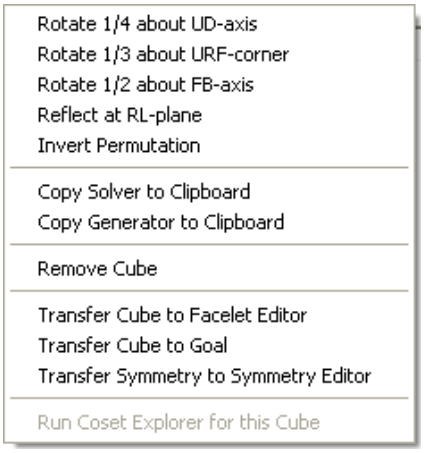
un carré apparaît autour de lui. Vous pouvez étendre la sélection à l'aide des touches Shift, Control Shift ou Control comme la manière habituelle.

•

Si vous déplacez la souris sur la **case à cocher Optimal** (**optimal checkbox)**lors de l'exécution du solveur optimale, vous pouvez voir le temps et le nombre de noeuds que Cube Explorer a calculé jusqu'à ce niveau de recherche dans l'arbre.



Par un clic droit sur l'extérieur de la zone de cubes, ce menu déroulant apparaître.
L'explication est donnée sur la page qui traite du Menu Edition (**Edit**).



Par un clic droit sur l'un des cubes, un menu Popup apparaît. Vous pouvez choisir de **supprimer, de pivoter ou de refléter** (**Remove, Rotate or Reflect**)le cube, ou de le transférer dans l'éditeur de figures pour une édition supplémentaire.
Si vous faites pivoter ou refléter un cube, vous pouvez ne pas obtenir le résultat attendu. La raison en est que le cube n'est pas seulement une tourné ou réfléchi, mais aussi recolorée,
de façon à ce que les couleurs du centre de ce cube restent toujours les mêmes.

Inverser Permutation (**Invert Permutation**) B donne le cube inverse du cube donné A.
Un générateur pour le cube A est un solutionneur de cube B et vice versa.
Copie Solveur / générateur dans le Presse-papiers(**Copy Solver/Generator to Clipboard**) permet de coller des manœuvres dans une autre application.

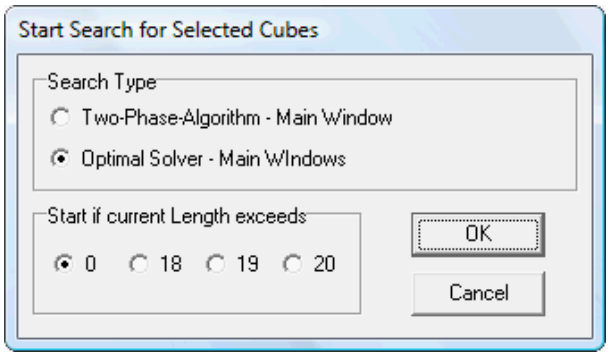
Lancer l'exploration de la la classe d'équivalences pour ce Cube (**Run Coset Explorer for this Cube**) est uniquement disponible dans la version spéciale 3 GB "s" de Cube Explorer. 278.691.840 classes d'équivalences définies sont résolues de façon optimale par un clique droit sur le cube dans un délai d'environ deux jours.
Cela signifie environ 1600 solutions optimales de cubes résolues par seconde!

Le Menu Démarrer

(The Run Menu)

Habituellement, vous lancez une recherche pour un cube en cliquant sur la petite flèche verte sur le côté droit de la Fenêtre principale. Si vous souhaitez exécuter la recherche pour de nombreux cubes en parallèle ou en séquentiel, vous pouvez utiliser ce sous-menu.

1. Lancer la recherche pour une sélection Cubes ...
(Start Shearch for Selected Cubes...)



Vous pouvez sélectionner un ou plusieurs cubes utilisant la touche Ctrl et / ou la touche Shift en même temps que la souris. Ces cubes sont résolus en parallèle avec les Deux-Phase-Solver ou Optimal Solver si la manœuvre de la longueur dépasse la valeur donnée par les boutons radio.

1. Stop Recherche pour une sélection Cubes
(Stop Search for Selected Cubes) Evidement cela arrête le recherche.

2. Préciser le nombre de pas auto …
(Define Maximum Number of Autorun Threads)

Cube Explorer prend en charge les processeurs multiples
pour calculer plus d'une manœuvre à la fois.
Définir le nombre de pas en parallèles ici

3. Démarrer Auto calcul (Autorun)pour le Solveur en deux passes

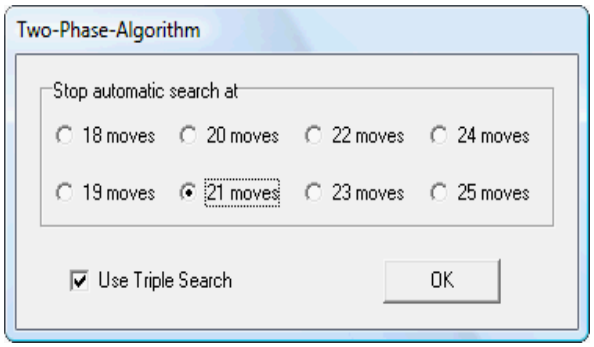
Ceci déclenche le Solver en deux phases ou le Solveur
Optimal qui commence avec le cube numéro 1 si aucuns cubes
n'est sélectionné, c'est le premier cube qui est sélectionné.
Stop le système de recherche automatique (**Stop Automatic Search at**)
valeur dans le le menu Options | Two-Phase-Algorithm....
affecte le comportement du module auto calcul deux phases.

4. Démarrer Autodémarrage pour le Solveur Optimal
(Start Autorun for Optimal Solver)

Le menu Options

(The Options Menu)

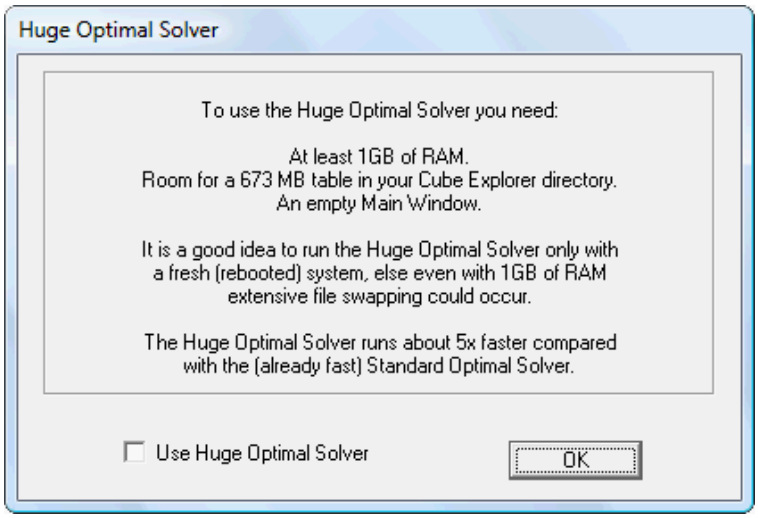
1. l'Algorithme en deux phases...
(Two-phase-Algorithm ...)



The Two-Phase-Algorithm ne calcule pas une seule solution, mais une série de solutions de manœuvres de plus en plus courtes. L'algorithme ne s'arrêtera pas tant que la longueur de manœuvre est supérieure à la valeur la de manœuvre donnée dans arrêt automatique de larecherche(**Stop automatic search at**).

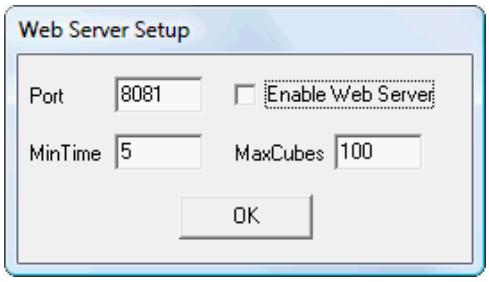
L'Algorithme en deux phases peut donner différentes solutions
dans les différentes orientations du cube.
Utilisez la Triple recherche (**Use Triple Search**) fait fonctionner l'algorithme de trois façons possibles
dans le même temps et donne la plus courte solution globale.
Dans la plupart des cas, cela accroît considérablement la performance de l'algorithme.

2. Le Solveur optimal en mode large...
(**Huge Optimal Solver ...**)



Pour exécuter l'Optimal Solver en mode large vous avez besoin de 1 Go de RAM. Un cube aléatoire cube sera résolu de façon optimale en moins d'une demie-heure en moyenne(PIII, 550MHz) . La case à cocher est désactivée si vous n'avez pas assez de RAM ou de l'espace sur votre disque dur.
Il y a aussi une version spéciale pour Cube Explorer qui est disponible si vous avez au moins 3 Go de Ram fonctionnant sous Windows XP Professionnel ou sur une version 64 bits de Windows. Elle est environ 15
fois plus rapide par rapport à la version Standard d'Optimal Solver.

3. Serveur Web...
   (Web Server ...)



Vous avez besoin de cette fonctionnalité seulement si vous voulez utiliser d'autres logiciels ou guider des robots avec Cube Explorer. Choisissez le port où le serveur est listé et validez le. Les détails comment utiliser cette interface sont décrites ici. Pour des raisons de sécurité, vous ne devez pas valider le Web Server si vous n'utilisez pas cette fonctionnalité.

4. Libérer immédiatement les pas
   (**Free Threads Immediately**)

Habituellement, vous pouvez arrêter le calcul d'une manœuvre sur un un cube et continuer plus tard car le pas de calcul est seulement arrêté mais pas supprimé ..
Si vous travaillez sur des milliers voir plus de cubes simultanément,les milliers de pas de fonctionnement peuvent créer des problèmes. Vous pouvez dans ce cas activer Libérer immédiatement les pas. L'inconvénient est que le calcul de manœuvres des cube recommence au début si vous continuez la recherche.

5. ignorer Isomorphiques à partir d'un fichier
   (**Skip Isomorphics when Loading from File**)

Lorsque vous ajoutez des cubes à partir d'un fichier dans la Fenêtre principale, vous avez la possibilité d'ignorer les cubes qui sont déjà dans la fenêtre principale d'isomorphie.

6. Inclure les Inversions Isomorphiques
   (**Isomorphy Includes Inversions**)

Vous pouvez inclure des inversions dans la définition des isomorphies. Cette fonctionnalité ne concerne pas seulement le chargement de cubes à partir d'un fichier, mais toutes les opérations où les fichiers sont ajoutés à la Fenêtre Principale, par exemple dans l'éditeur de

figures (Pattern Editor) ou l'éditeur de symétries (Symmetry Editor.).

7. Utiliser la rotation de faces (**Use Center Facelet Twists**)

S'il ya des chiffres (par exemple, Sudoku Cube) ou photos dans le centre des faces, la rotation des centres sont importante. Cela augmente la taille de l'espace du cube d'un facteur 2048. Pour une utilisation de Algorithme en deux phases,une grande taille de tableau est nécessaire, il est créé dès le premier tour.
Vous devez avoir 1 Go de RAM pour utiliser cette fonction.

Note: The Optimal Solutionneur peut-être trop lent dans de de nombreux cas, pour résoudre un cube avec les centres mélangés en un temps concevable. Lancerde l'Optimal Solver en mode large n'augmentera pas la vitesse de ce genre de problème.

8. Modifier la rotation des faces centrales **Edit Center Faclet Twists)**

Une boîte de dialogue apparaît où vous pouvez modifier les dissonances du centre . Le bouton Ok-est grisé si la la configuration de rotation est invalide.

9. Générer des statistiques (**Generate Statistics**)

Quelques informations sur la durée de la manœuvre dans la fenêtre principale, les manœuvres optimales et le nombre de cubes antisymétriques est affiché dans une fenêtre séparée.

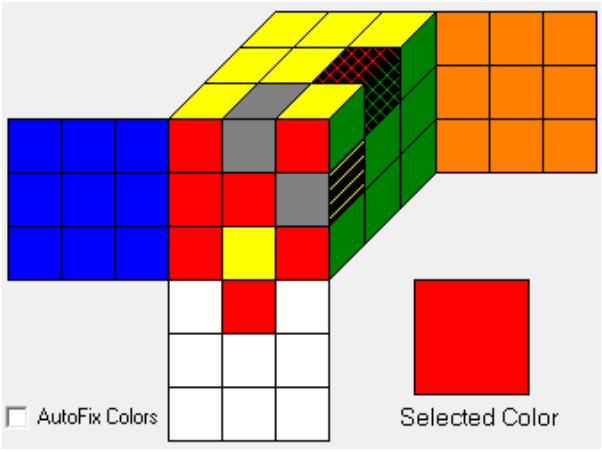
Résoudre les cubes incomplets

(Solving incomplète Cubes)

Depuis la version 4.0 Cube Explorer est capable de résoudre les cubes incomplets.
Dans l'éditeur de faces (Facelet-Editor) une face grise désigne une face indéfinie.
Avec le bouton droit de la souris,vous pouvez undéfinire des faces.
Si les deux d'un faces d'un bord ou les trois faces d'un coin ne sont pas définis,
le bord ou le coin n'est pas défini et peut avoir n'importe quelles positions et orientations dans le cube résolu.
L'utilisation de la du bouton gauche avec la touche Ctrl ou la touche Maj donne deux autres possibilités pour définir des faces-voir ci-dessous.

Si vous résoudre un cube incomplet de en sortie, habituellement de nombreuses solutions sont affichées dans une fenêtre séparée.
Vous pouvez modifier la sortie et la copier (ctrl-c) et coller (Ctrl-V) dans un éditeur de texte de votre choix.

Il y a **quatre** possibilités pour les couleurs des coins / bords, qui sont expliqués par l'exemple suivant:



1. Le bord de cube de la position FD avait comme couleurs normale jaune et rouge. Après la résolution de ce cube,
le bord de cube aura comme position d'origine la position UF avec une orientation correcte.

2. Le bord de cube a deux face grisée dans la position UF. Après la résolution de ce cube, les petits cubes ont
position et une orientation quelconque dans le cube.

3. Le bord de cube de la position UR a deux faces hachurées en couleurs rouge et verte. Après avoir résolu
ce cube, le petit cube est dans sa position finale, mais l'orientation peut-être arbitraire. Utilisez la touche Majuscule
et le bouton gauche de la souris pour entrer les faces hachurées .

4. Le bord de cube de la position le FR a une face en gris et une en jaune avec un hachurage. Après la résolution
de ce cube, le petit cube a la bonne orientation (définie par le hachurage diagonal), mais la position peut-être
arbitraire. Utilisez la touche Ctrl avec le bouton gauche de la souris pour entrer les faces hachurées.

Si vous appuyez sur le bouton **Ajouter et de résoudre(Add and Solve),** les solutions possibles seront répertoriés dans une fenêtre séparée.
Cette fenêtre a un bouton pour ajouter les solutions à la fenêtre principale (Main windows).
Vous voyez tous les cubes correspond aux définitions incomplètes du cube de l' éditeur de faces dans cette fenêtre principale.

L'algorithme utilisé pour les cubes incomplèts est beaucoup plus lent que les deux phases de l'algorithme pour les cubes complets.
Néanmoins, il est utile de trouver des manœuvres jusqu'à environ 15 coups, en fonction de la situation.

Si vous ne voulez pas tourner certaines faces du cube, vous avez la possibilité de les exclure (**exclude)**de la recherche.
Soyez conscient que vous pourriez ne pas avoir de solution du tout en excluant trop de faces!

Permettre la rotation initiale du cube (**Allow Initial Cube Rotation**)signifie que Cube Explorer essaie de trouver
d'autres orientations du cube où les restrictions de votre cube sont respectées.
La solution de chaîne UL 'F R2 (15f) / / préfixe C_R2U par exemple, signifie que vous déplacez le **cube entier** comme avec R2 et U
en premier (vous touchez le cube avec une seule main tout en tournant!), puis vous appliquez UL 'F R2.
Ensuite, vous revenez à la position de cube original en appliquant C_U'R2 (l'inverse de C_R2U).

Ignorer la rotation des centres
(**Center Twists To Ignore**) est grisé dans le mode standard de Cube Explorer.
Si vous avez Options | Utilisation rotation des centres(**Options|UseCenter Facelet Twists**)
cochée, vous pouvez ignorer certaines rotations de centres en cochant la case à cocher correspondante ici.

L'Interface de Cube Explorer

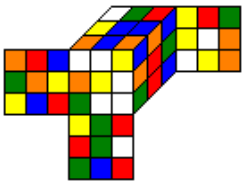
(The Cube Explorer Interface)

Vous voulez construire un robot de résolution de cube que vous souhaitez utiliser avec Cube Explorer pour trouver la solution
des manœuvres?
J'ai décidé de mettre en œuvre un serveur simple, qui connecté en l'écoute sur un port de votre choix.

Cas A: Vous voulez que Cube Explorer vienne de calculer les manœuvres pour vous

Si vous avez activé le serveur Web, par exemple sur le port 8080 dans les **Options | Web Server ...** menu, vous pouvez envoyer une chaîne de caractères à ce port, qui encode les positions des 54 faces. Vous pouvez envoyer cette chaîne de caractères en utilisant votre selfwritten logiciel ou d'un simple navigateur. Voici un exemple pour le port 8080 sur le port local(localhost):

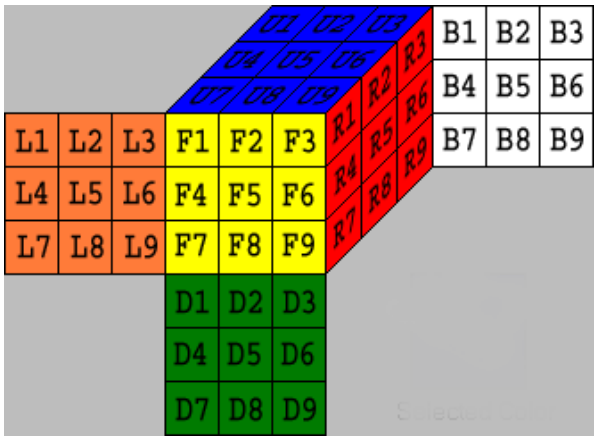
http://127.0.0.1:8080/?bdrfuulululululrddrubbflfbdbbfdrrdbdurlrudlffurfrdfblbfl



Le cube qui est codée par la chaîne ci-dessus.

Tous les caractères envoyée avant le "?" sont ignorés. Par exemple dans le cube la couleur bleu est représenté au-dessus face du centre U, il faut écrire "u" dans toutes les positions des facettes bleu. Les positions sont définies dans l'ordre suivant:
U1, U2, U3, U4, U5, U6, U7, U8, U9, R1, R2, R3, R4, R5, R6, R7, R8, R9, F1, F2, F3, F4, F5, F6, F7, F8, F9
D1, D2, D3, D4, D5, D6, D7, D8, D9, L1, L2, L3, L4, L5, L6, L7, L8, L9, B1, B2, B3, B4, B5, B6, B7, B8, B9

Ignorer les couleurs de la gauche du cube, cela montre quelle position correspond à telle facette.



L'exemple de caractères ci-dessus commence par "BDR,
parce que dans la position U1 du cube en exemple, nous avons la couleur de la face B en position U2 nous avons la couleur de la face D, en position U3 , nous avons la couleur de la face R. Continuez jusqu'à ce que la position B9 de cette façon.

Il y a plusieurs paramètres qui influent sur le temps de calcul et de la durée de la solution. Tout d'abord, il faut fixer la recherche automatique (**Stop Automatic Recherche**) et celle de la longueur maximale des manœuvres (**Maximal Maneuver Length**)dans les paramètres **Options | Two-phase Algorithm ... avec la même valeur** lors de l'utilisation de l'interface Web, par exemple à 21 ou 22 mouvements si vous
voulez une réponse rapide (la valeur par défaut est de 20 coups). La résolution ne donnera jamais plus de manœuvres que cette valeur, quel que soit le temps de calcul. Faites donc attention à ce paramètre et ne le fixez pas à moins de 19 car certaines positions ont besoin de 20 coups et que donc le serveur pourrait

jamais donner la solution.

Dans l'exemple ci-dessus, la chaîne

U2 B U2 L2 DL 'B' D 'L' B2 D 'BUD' F2 U 'L2 U' F2 U F2 (21f)

est retourné.

Le paramètre TempsMin (**MinTime**) dans le menu **Options | Web Server ...** menu est le minimum de temps en secondes donné à Cube Explorer pour la recherche de solutions. Le temps de recherche actuel donne le maximum **de MinTime** et du temps de trouver une manœuvre de Durée maximale des manœuvres(**Maximal Maneuver length**).
Réglez la longueur maximale des manœuvres à 22 si vous voulez que le temps de recherche soit le même que dans MinTime dans presque tous les cas.

Le paramètre **MaxCubes**, indique combien de cubes au plus doivent stockés en interne et figurer dans la fenêtre Principale.
Ce paramètre est seulement important si vous exécutez le Cube Explorer Web comme un service sans interface graphique (par exemple, avec l'aide de l'utilitaire Srvany.exe).

Cas B: Vous voulez utiliser la WebCam Interface à la numérisation des facettes du Cube

Ok, votre robot ne semble pas avoir les yeux. Alors permettez à Cube Explorer faire ce travail pour vous.

Vous pouvez contrôler le processus de numérisation du cube par l'interface de la webcam par l'envoie de chaînes de carractère sur Serveur Web(WebServer).
Tout d'abord configurez tous les paramètres de la webcam interface, pour qu'il fonctionne dans un mode "manuel" fiable.

Maintenant vous pouvez positionner la face arrière du cube en face de la webcam avec votre robot (comme vous le feriez manuellement) et de l'envoyer par exemple la chaîne "http://127.0.0.1:8080/?scanB" WebServer pour scanner la face arrière du cube.
Ensuite, vous placez la face gauche et envoyer la chaîne «http://127.0.0.1:8080/?scanL".
Procéder de cette façon, en utilisant les chaînes "? Scanf", "? ScanR", "? Scanu" et "? SCAND". Le serveur Web répondra par un "Fait!"("Done!") dans tous les cas.

Au lieu d'appuyer sur résoudre Cube numérisée(**Solve Scanned Cube**), vous envoyez la chaîne« http://127.0.0.1:8080/?transfer ".
Si toutes les faces sont ok, la numérisation du cube est transféré dans la fenêtre principale et la résolution des manœuvres est calculée. Le serveur Web répond avec un "Done!", Et non pas manœuvre calculée!

Enfin, avec la chaîne "http://127.0.0.1:8080/?getLast" le serveur répond avec la dernière manœuvre résolue calculée dans la fenêtre principale de Cube Explorer. Cette chaîne met en route votre robot pour résoudre le cube.

La chaîne de caractères "? Getlast" est également utile, si vous entrez manuellement les couleurs des faces et que voulez que votre robot les tournes.

La commande "? Clair" efface la fenêtre principale.

La commande "? Connect0" se connecte à la Web Cam, si vous ne voulez pas le faire manuellement dans le Cube Explorer.
Si vous avez plusieurs périphériques de capture connecté à votre PC, "? Connect1" se connecte au second dispositif, "? connect2" à la troisième périphérique, etc.

Le commande "? Statut" renvoie les couleurs des faces déjà numérisées en mode interactif comme le montre le Scan de prévisualisation (Scan Preview). Une face grise est retourné comme «x» et l'ordre des faces est expliqué ci-dessus.
Exemple:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxoggrrrorrwwbybyyby

Je n'ai jamais construit un robot moi-même. Si vous découvrez une caractéristique oubliée ou avez d'autres suggestions, s'il vous plaît faites le-moi savoir.

Utilisation de l'Aide-System

(Using the Help-System)

s

Vous pouvez accéder à l'aide en appuyant sur F1 ou par «Aide | Table des matières» ("Help|Contents")dans le menu

Attention !!

Certaines formules sont encore en Anglais pour celle qui sont traduites il vaut mieux lire la formule originale sur le fichier d'aide Cube464.chm rédigé par l'auteur

Tout ceci afin de minimiser les erreurs

Introduction

Les pages qui suivent sont une tentative pour donner un aperçu des idées mathématiques et des algorithmes développés et utilisés dans Cube Explorer.

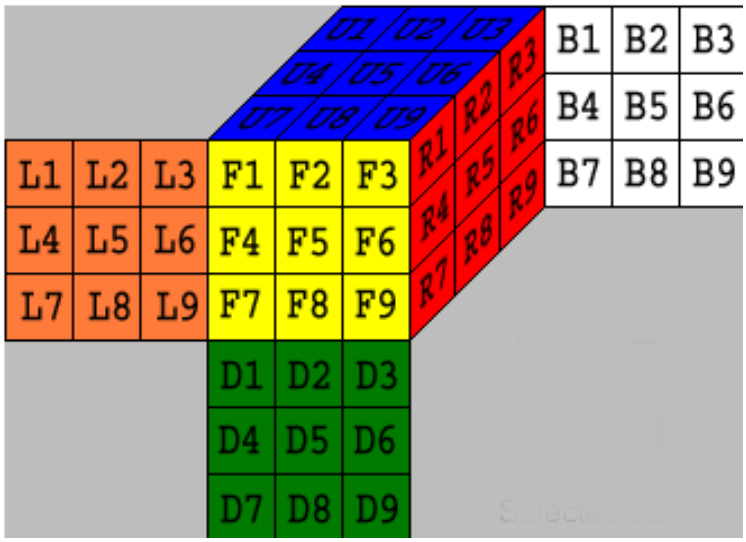
J'ai eu à me battre avec plusieurs problèmes . Tout d'abord, l'anglais n'est pas ma langue maternelle et de certaines de mes explications pouvaient être difficiles à comprendre ou incompréhensible à tous. Ensuite, j'ai étudié les mathématiques Il y a longtemps et ma terminologie sera certainement incorrect dans certaines parties. Troisièmement, je ne pourrait esquisser que les grandes idées de tous que ce qui a été nécessaire pour écrire Cube Explorer.

Mais j'espère que, néanmoins, cela sera une aide pour ceux qui sont intéressés à comprendre l'Algorithme en deux Phases (**Two-Phases Algorithm)**ou veulent mettre en œuvre l'algorithme de leur propre programme.

Permutations et niveau de Facettes

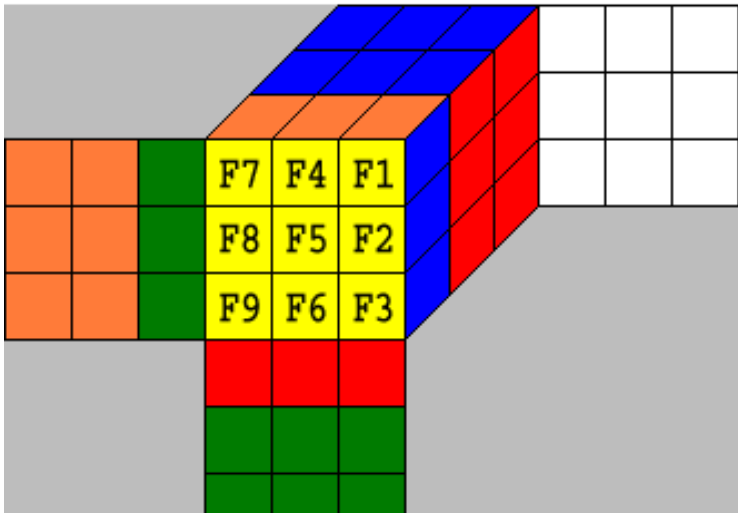
(Permutations and the Facelet Level)

Si nous regardons un cube propre, on voit 6 * 9 facettes.



Si l'on applique un mouvement au cube, les facettes sont réarrangées. Cette réorganisation est nommée permutation.
Nous utilisons les six lettres U, R, F, D, B, L pour décrire les six mouvements à 90 ° des faces vers la droite. Nous utilisons F2 pour par exemple indiquer une rotation à 180 ° et F' pour indiquer une rotation à 270 °, c'est-à-dire une rotation 90 ° sens anti-horaire de la face avant.

Si l'on applique par exemple une rotation F au cube représenté ci-dessus, nous obtenons les résultats suivants:



Pour expliquer la représentation de ces permutations, nous allons seulement regarder les facettes jaunes pour un moment.
Il existe deux possibilités pour cette représentation dans l'exemple.

1.
F1 est amené à (is carried to)F3 (F1-> F3), F2-> F6, F3-> F9, F4-> F2, F5-> F5, F6-> F8, F7-> F1, F8-> F4, F9-> F7. Nous pouvons
écrire
F1 F2 F3 F4 F5 F6 F7 F8 F9
F3 F6 F9 F2 F5 F8 F1 F4 F7

2.
F1 est remplacé par (is replaced by)F7 (F1 <-F7), F2 <-F4, F3 <-F1, F4 <-F8, F5 <-F5, F6 <-F2, F7 <-F9, F8 <-F6, F9 <-- F3. Nous pouvons écrire
F1 F2 F3 F4 F5 F6 F7 F8 F9
F7 F4 F1 F8 F5 F2 F9 F6 F3
Parce que la première rangée de tables est toujours le même, on peut supprimer cette ligne. Ainsi, nous pouvons écrire
(F3, F6, F9, F2, F5, F8, F1, F4, F7) dans est amené à de la représentation ou (F7, F4, F1, F8, F5, F2, F9, F6, F3) dans est remplacé par la représentation.
Dans la plupart des cas nous ne pourrons pas utiliser la forme abrégée, sans un tableau comme ici pour des raisons de clarté.
Nous utilisons la première représentation sur le facelet niveau(arrêtes et coins), et la seconde sur les cubie niveau(cubes centraux).
Dans le reste ce chapitre c'est "amené à" qui est utilisé comme représentation.

Nous sommes en mesure de définir un produit de deux permutations.
Par exemple
F1 F2 F3 F4 F5 F6 F7 F8 F9
F2 F1 F6 F3 F5 F4 F8 F7 F9
*
F1 F2 F3 F4 F5 F6 F7 F8 F9
F3 F6 F9 F2 F5 F8 F1 F4 F7

=

F1 F2 F3 F4 F5 F6 F7 F8 F9
F6 F3 F8 F9 F5 F2 F4 F1 F7

parce que, par exemple F1-> F2 par la première permutation et F2-> F6 par la seconde, nous avons donc F1-> F6 dans le produit.

La multiplication de permutation a quelques similitudes avec la multiplication avec des nombres, mais il y a une grande différence: alors que pour l'exemple 3 * 5 = 5 * 3, vous avez généralement pas à échanger l'ordre de deux permutations.

Mais dans l'exemple ci-dessus, nous avons
F1 F2 F3 F4 F5 F6 F7 F8 F9
F3 F6 F9 F2 F5 F8 F1 F4 F7
*
F1 F2 F3 F4 F5 F6 F7 F8 F9
F2 F1 F6 F3 F5 F4 F8 F7 F9
=
F1 F2 F3 F4 F5 F6 F7 F8 F9
F6 F4 F9 F1 F5 F7 F2 F3 F8
et c'est quelque chose de différent. La multiplication des permutations n'est pas commutative.

Un autre terme important est la permutation inverse.
Prenons le mouvement F
F1 F2 F3 F4 F5 F6 F7 F8 F9
F3 F6 F9 F2 F5 F8 F1 F4 F7
et la permutation
F1 F2 F3 F4 F5 F6 F7 F8 F9
F7 F4 F1 F8 F5 F2 F9 F6 F3
Dans ce cas,
F1 F2 F3 F4 F5 F6 F7 F8 F9
F3 F6 F9 F2 F5 F8 F1 F4 F7
F1 F2 F3 F4 F5 F6 F7 F8 F9
F7 F4 F1 F8 F5 F2 F9 F6 F3

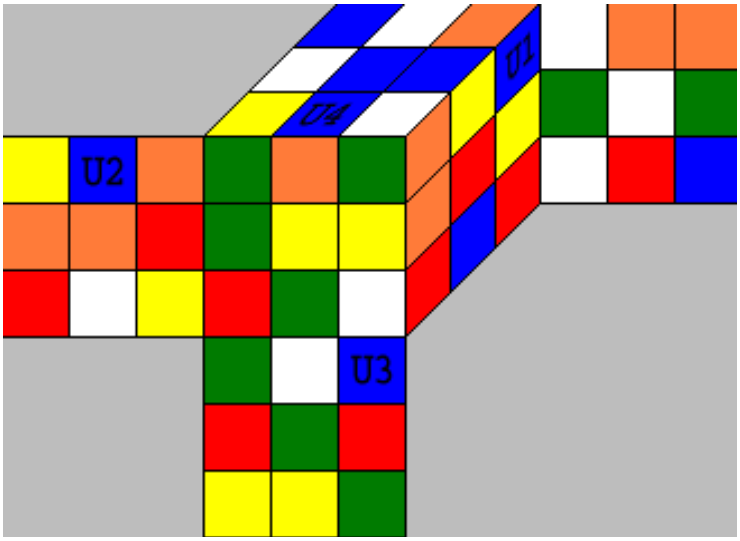
=

F1 F2 F3 F4 F5 F6 F7 F8 F9
F1 F2 F3 F4 F5 F6 F7 F8 F9

Cette permutation ne fait rien du tout, quand on multiplie une permutation avec son inverse, on obtient le permutation à identité I. En fait, dans cet exemple la deuxième permutation représente de F', donc nous avons F * F '= I, qui est tout à fait évident.
Vous pouvez vérifier aussi que F '* F = I, donc dans ce cas particulier, la multiplication est commutative.

Dans le fichier CubeDefs.htm, vous pouvez voir la définition complète des mouvements de base. Par exemple

F: = (U1, U2, U3, U4, U5, U6, R1, R4, R7, D3, R2, R3, D2, R5, R6, D1, R8, R9,
F3, F6, F9, F2, F5, F8, F1, F4, F7,
L3, L6, L9, D4, D5, D6, D7, D8, D9, L1, L2, U9, L4, L5, U8, L7, L8, U7, B1, B2,
B3, B4, B5, B6, B7, B8, B9),
écrit dans la forme courte, sans table.
Ce ne sont pas seulement les mouvements qui peuvent être considérés comme des



permutations, mais tous les cubes codé peuvent être écrit comme une permutation.

Parce que U1-> R3, U2-> L2, U3-> D3, U4-> U8, ... ce cube a la représentation (R3, L2, D3, U8 ,...).

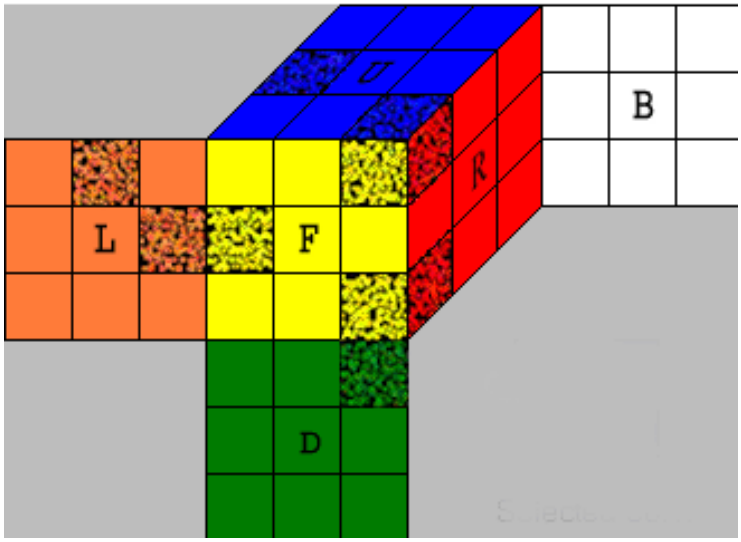
Si vous résolvez ce cube, vous, en fait, vous essayez de trouver la permutation inverse de ce cube composé comme un produit des permutations correspondant aux mouvements élémentaires U, U2, U, R, R2, R'..... Rappelons que le produit d'une permutation avec la permutation inverse donne l'identité de permutation, et c'est le cube propre. L'algorithme de résolution de de Cube Explorer essaie de trouver les plus courts produits pour cet inverse.
Pour l'image d'exemple ci-dessus, il constate, en quelques secondes

R2 * L * U2 * L2 * D * R2 * U2 * L '* D2 * R * U * B * R * F2 * L2 * B2 * L2 * B2

Mais représenter les permutations au niveau des facettes n'est pas efficace pour un calcul rapide. Il y a deux autres niveaux pour se débrouiller.

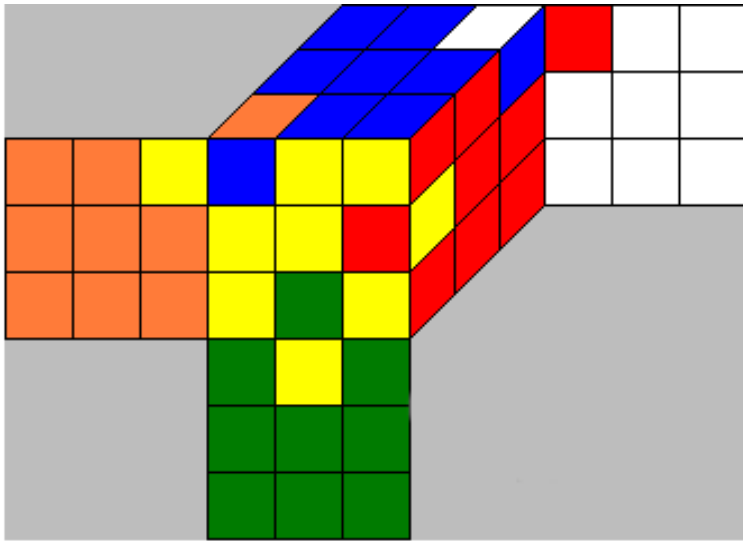
Le niveau Cubie (cube du centre)
(The Cubie Level)

Sur le niveau cubie, les objets que nous permutons ne sont pas les facettes, mais les 12 arêtes et 8 coins.



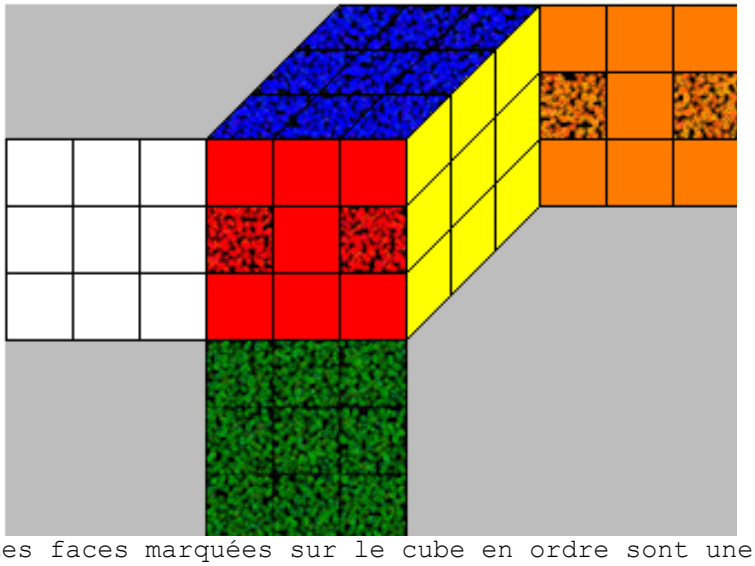
Dans l'image ci-dessus, le coin-URF, le coin-DFR, le bord-FL et l'arête-UL sont marqués.
Les coins sont nommés URF, UFL, ULB, UBR, DFR, DLF, DBL et DRB. Les bords sont nommés UR, UF,
UL, UB, DR, DF, DL, BD, FR, FL, BL et BR.

Sur le niveau cubie, il n'est pas possible de représenter un mouvement ou d'un cube brouillé par une simple permutation, parce que les coins peuvent être tournés et les arrêtes peuvent être permutées.

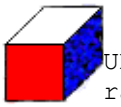


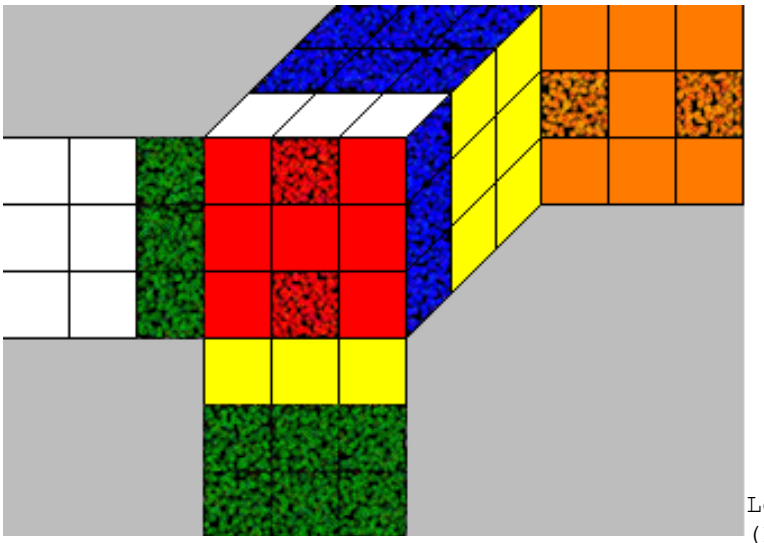
Ici, les cubies sont dans leurs positions, mais les orientations ont changé. Le coin UFL à tourné dans le sens des aiguilles d'une montre, le coin-UBR a tourné le sens anti-horaire et les arrêtes DF et RF ont été permutées.

Si les coins ou les bords ne sont pas dans leur positions, il existe de nombreuses façons de définir les orientations des cubies.
Mais pour le deux phases-Algorithm, la définition suivantes est nécessaire.



Les faces marquées sur le cube en ordre sont une référence pour l'orientation.

Dans l'image, le coin  URF est tourné sens des aiguille d'une montre (vers la droite) par rapport à la référence sur la face en ordre du cube,de même pour le coin DLF. Les coins UFL et DFR sont tournés vers la gauche.
Les arrêtes UF, DF, FL FR ont été permutées.



Le Mouvement-F
(F-move)

Nous utilisons la représentation "est remplacé par" pour décrire les permutations sur les Cubie niveau. Dans cet
exemple: URF est remplacé par UFL (URF <-UFL), UFL <-DLF, ULB <-ULB, UBR <-UBR, DFR <-URF, DLF <-DFR, DBL <-DBL, DRB <-DRB.
Nous écrivons
URF UFL ULB UBR DFR DFL DBL DRB
UFL DLF ULB UBR URF DFRDLB DRB
pour la permutation des coins dans ce cas.

Mais nous avons aussi de suivre l'évolution des orientations, et nous écrivons

F =

URF	UFL	ULB	UBR	DFR	DLF	DBL	DRB
c:UFL;o:1	c:DLF;o:2	c:ULB;o:0	c:UBR;o:0	c:URF;o:2	c:DFR;o:1	c:DBL;o:0	c:DRB,o:0

Nous utilisons des «0» si la rotation ne change pas, "1" pour un tourné à droite et "2" pour un tour en sens anti-horaire.

Ainsi, nous pouvons ajouter des orientations de manière simple. Si nous ne le faisons par exemple, deux rotations anti-horaire , le résultat
de rotations est 2 +2 = 4, car 4 = 1 mod 3, le résultat est une rotations sens des aiguilles d'une montre. Jetez un oeil sur CubeDefs.htm pour la définition des mouvements de base. La permutation des arêtes est définit similairement, avec "1" pour la permutation des arrêtes et de "0" pour une non permutation des arrêtes.

Nous avons également besoin d'une notation pour décrire une permutation sans l'aide d'une table.
Pour le mouvement-F ci-dessus de nous écrivons par exemple

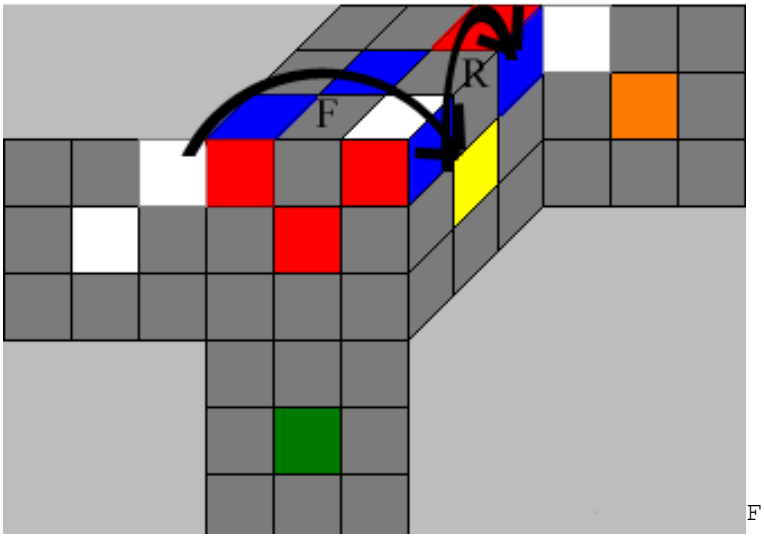
F (URF). C = UFL
F (URF). O = 1
F (UFL). C = DLF
F (UFL). O = 2
etc

Nous utilisons un autre mouvement

R =

URF	UFL	ULB	UBR	DFR	DLF	DBL	DRB
c:DFR;o:2	c:UFL;o:0	c:ULB;o:0	c:URF;o:1	c:DRB;o:1	c:DLF;o:0	c:DBL;o:0	c:UBR,o:2

pour montrer comment définir le produit F * R de ces deux permutations, orientations comprises.



F * R appliqué au coin-UFL

F (URF).c= UFL et F (URF). O = 1 le tableau ci-dessus nous dit que l'angle de position URF est remplacé par l'angle de position UFL et que l'orientation du coin qui se déplace sur la position URF est augmenté de 1 lors de l'exécution d'un déplacement-F.
R (UBR). URF et c = R (UBR). O = 1 nous dit que l'angle à la position UBR est remplacé par l'angle de URF position et que l'orientation augmente de 1 lors de l'exécution d'un déplacement-R.

Ainsi, lors de l'exécution de F * R, nous avons URF <-UFL par le déplacement-F, puis UBR <-URF par le déplacement-R, il en résulte que UBR <-UFL. Alors nous

avons le résultat (F * R) (UBR). C = UFL. Cela signifie que (F * R) (UBR). C = F (R (UBR). C). C.

Le comportement de l'orientation est plus difficile à comprendre. F nous dit cela F (URF). O = 1 lorsque URF <-UFL.
Cela signifie, l'orientation du coin qui se déplace de la position UFLvers la la position URF augmente de 1.
Cette orientation s'ajoute à la modification de l'orientation du coin qui se déplace de l'URF vers l'UBR (UBR <-URF) du déplacement-R . Nous avons donc F (URF). O + R (UBR). O pour les changement d'orientation
de la position UBR, et parce que URF = R (UBR). c, nous avons (F * R) (UBR). o = F (R(UBR). C). O + R (UBR). O.

En général, pour deux permutations A et B et pour toute position x d'angle, nous avons:
(A * B) (x). C = A (B (x). C). C
et
(A * B) (x). O = A (B (x). C). O + B (x). O

Le même principe vaut pour les permutations de pointe. Voir CubeDefs.htm pour la mise en œuvre de la routines de multiplication.
Si nous voulons aussi à inclure le cas de la réflexion, dont nous avons besoin si nous appliquons les symétries du cube, les choses sont un peu plus compliquées avec les orientations des coins. Au lieu d'ajouter modulo 3 dans la deuxième équation ci-dessus, qui peut être interprété comme un groupe d'opération dans le groupe cyclique C3, et nous travaillons dans le groupe dièdre D3. Nous décrivons les trois éléments supplémentaires dans ce groupe avec les numéros 3, 4 et 5.

Le Niveau Coordonnée

(The Coordinate Level)

Le niveau de coordination décrit les permutations et les orientations des bords et des coins avec les chiffres naturels.
Ce niveau d'abstraction est particulièrement adapté pour mettre en œuvre un algorithme rapide à résoudre le cube.

La définition des coordonnées d'orientation de l'angle (ou coins)
(The definition of the corner orientation coordinate)

Si l'on applique par exemple le R pour la remise en ordre du cube nous avons

URF	UFL	ULB	UBR	DFR	DLF	DBL	DRB
c:DFR;o:2	c:UFL;o:0	c:ULB;o:0	c:URF;o:1	c:DRB;o:1	c:DLF;o:0	c:DBL;o:0	c:UBR,o:2

L'orientation des 8 coins est décrites par un certain nombre de
0 à 2186 (3 ^ 7 -1).

En cubicube.pas vous trouverez la définition suivante

```
function CubieCube.CornOriCoord:Word;
var co: Corner; s: Word;
begin
  s:=0;
  for co:= URF to Pred(DRB) do s:= 3*s + PCorn^[co].o;
  Result:= s;
end;
```

Dans l'exemple ci-dessus, cette fonction calcule
2 * 3 ^ 6 + 0 * 3 ^ 5 + 0 * 3 ^ 4 + 1 * 3 ^ 3 +1 * 3 ^ 2 + 0 * 3 ^ 1 + 0 * 3 ^ 0 = 1494
C'est juste le nombre 2001100 dans un système de numérotation ternaire.

Pour rendre cette méthode de travail facile, nous devons l'écrire dans la représentation de la permutation dans "est remplacé par".
elle ne fonctionnent pas dans la représentation "est déplacé vers"!
Nous ignorons à l'orientation du coin-DRB, parce que cette orientation est déterminée par l' orientation des
les sept autres angles: la somme de toutes les orientations doit être divisible par trois.

La définition des coordonnées de l'orientation arrêtes (ou bords)

(The definition of the edge orientation coordinate)

La coordination de l'orientation des bords est définie de manière analogue.

L'orientation des 12 arêtes est décrit par un nombre de 0 à 2047 (2 ^ 11 -1).

dans cubicube.pas vous trouverez la définition suivante

```
function CubieCube.EdgeOriCoord:Word;
var ed: Edge; s: Word;
begin
  s:=0;
  for ed:= UR to Pred(BR) do s:= 2*s + PEdge^[ed].o;
  Result:= s;
end;
```

Nous utilisons ici le système binaire au lieu du système ternaire . Nous ignorons à l'orientation des bords-BR,
car elle est déterminée par l'orientation des 11 autres bords. La somme de toutes les orientations doit être pair.

La définition des coordonnées de permutation coins
(The definition of the corner permutation coordinate)

Les coordonnées de permutation des coins sont données par un nombre de 0 à 40319 (8! -1).
Dans cet exemple, nous utilisons encore la permutation déplacer-R, mais l'on ignore maintenant les orientations.

URF	UFL	ULB	UBR	DFR	DLF	DBL	DRB
c:DFR	c:UFL	c:ULB	c:URF	c:DRB	c:DLF	c:DBL	c:UBR
	1	1	3	0	1	1	4

Nous définissons l' ordre naturel des angles par URF <UFL <ULB <UBR <DFR <DLF <DBL <DRB.
Le nombre dans la troisième ligne sous le coin XXX de la deuxième ligne indique le nombre de tous les coins à la gauche de XXX, dont l'ordre est plus élevé que l'ordre de XXX.
Au-dessus de l'entrée 4, nous avons par exemple le coin UBR.
Depuis les 7 coins à gauche UBR, 4 coins on un ordre supérieur, DFR, DLF, DBL, DRB.
au-dessus de l'entrée 1, nous avons par exemple le coin DLF.
Depuis les 5 coins gauche de DLF, 1 coin a un ordre supérieur DRB.
Nous construisons les coordonnées de la permutation avec les numéros de la troisième rangée.
1 * 1! + 1 * 2! + 3 * 3! + 0 * 4! + 1 * 5! + 1 * 6! + 4 * 7! = 21021

Donc, nous utilisons une base un système variable ici.

La fonction suivante à partir de cubicube.pas fait le travail.

```
function CubieCube.CornPermCoord: Word;
var i,j: Corner; x,s: Integer;
begin
  x:= 0;
  for i:= DRB downto Succ(URF) do
  begin
    s:=0;
    for j:= Pred(i) downto URF do
    begin
      if PCorn^[j].c>PCorn^[i].c then Inc(s);
    end;
    x:= (x+s)*Ord(i);
  end;
  Result:=x;
end;
```

La définition de coordonnées de permutation des arrêtes (bords)
(The definition of the edge permutation coordinate)

Les coordonnées de permutation des bords sont définies de la même façon de 0 à 12!-1.

nous utilisons la function suivante dans Cubicube.pas

```
function CubieCube.EdgePermCoord: Integer;
begin
  x:= 0;
  for i:= BR downto Succ(UR) do
  begin
  s:=0;
  for j:= Pred(i) downto UR do
  begin
    if PEdge^[j].e>PEdge^[i].e then Inc(s);
  end;
  x:= (x+s)*Ord(i);
  end;
  Result:=x;
end
```

Maintenant, nous sommes en mesure de décrire chaque cube avec et (x1, x2, x3, x4) et
0 <= x1 <3 ^ 7, 0 <= x2 <2 ^ 11, 0 <= x3 <8!, 0 <= x4 <12!

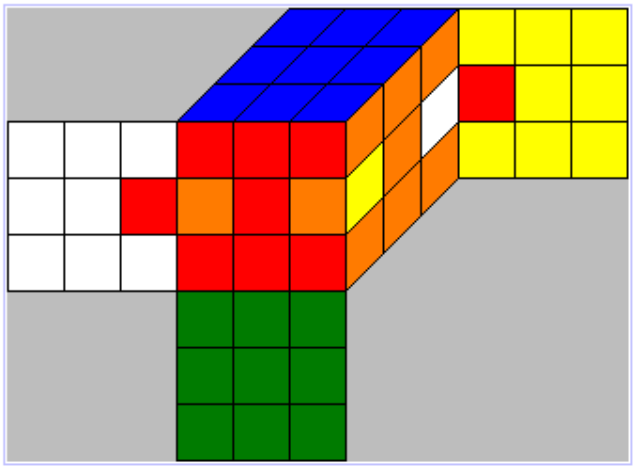
Seulement la moitié de ces cubes est vraiment possible à faire parce que toutes les permutations réalisables sont pair
nous ne disposons que de 12! * 8! / 2 permutations (en ignorant les orientations).

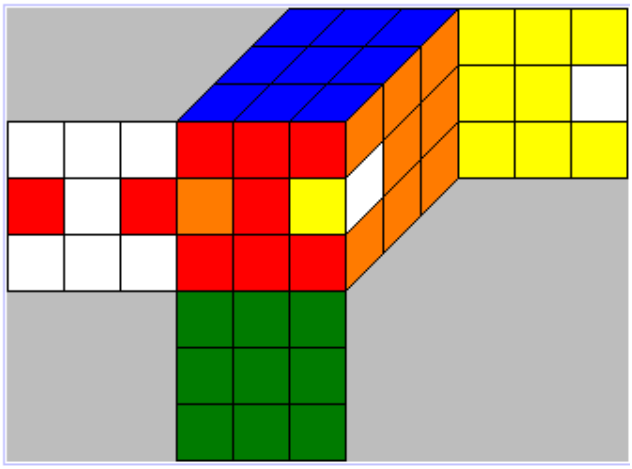
Le nombre de cubes est donc donnée par
3 ^ 7 * 2 ^ 11 * 8! * 12! / 2 = 43.252.003.274.489.856.000

Il y a d'autres coordonnées que nous utilisons pour l'Algorithme en deux phases nous verrons plus tard plus tard ..

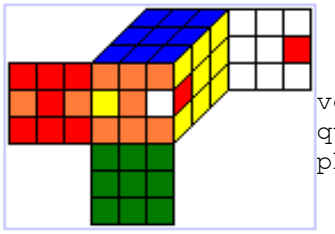
Cubes Équivalents et Symétrie

(Equivalent Cubes and Symmetry)





Regardez les deux cubes ci-dessus. Ils ont une apparence différente, mais au fond, ce sont les mêmes. Si vous tournez le cube à l'image de gauche de 90 degrés autour d'un axe passant par le centre-U et le centre D des cubies,

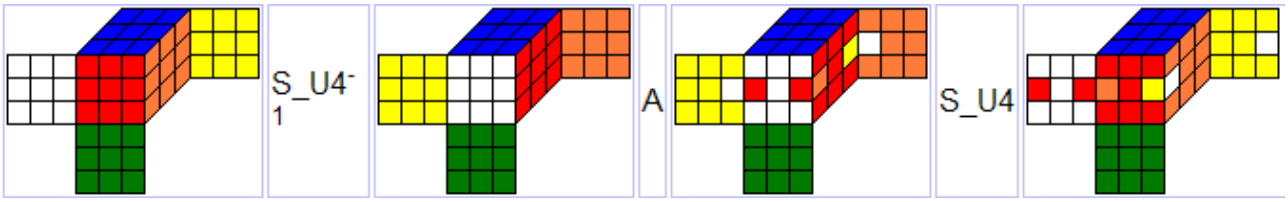


vous obtenez ce cube, et si vous recolorez la arrêtes pour que la face-F soit rouge etc vous obtiendrez le cube de la photo de droite. Nous appelons ces deux cubes équivalent.

Parce que les cubes équivalents ont la même structure, le nombre de coups nécessaires pour les résoudre est le même.

Définir des cubes équivalent à l'aide du recoloriage de faces n'est pas vraiment ce que nous voulons, parce que nous retournons au même niveau. Nous préférons la définir l'équivalence au niveau des permutations des cubie . Avec S_U4 on note la rotation à 90 degrés grâce au centre-U et au centre-D des cubies de l'ensemble du cube. cela désigne définit le cube ci-dessus à gauche avec A et la permutation pour le cube ci-dessus à droite cube avec B.

Alors, nous avons



Donc, nous avons

$$B = S^{-1} * A * S$$

dans cet exemple. En général, deux permutations de cube A et B sont équivalentes, s'il y a une symétrie S de la cube avec B = S-1 * A * S

Pour chaque cube, il y a jusqu'à l'équivalent de 48 cubes, parce que le cube a 48 symétries, y compris les réflexions.
Dans Cube Explorer, ces 48 symétries sont générés par quatre symétries "basiques":

S_URF3, tourne de 120 degrés autour du cube sur un axe passant par les coins UDF et de DBL,
S_F2, tourne à 180 degrés autour du cube sur un axe passant par les centres F et B,
S_U4, tourne de 90 degrés autour du cube sur un axe passant par les U et D
S_LR2, est une réflexion du plan de la tranche RL.

Ces symétries reflètent les permutations de base de la permutation des coins et des bords et sont décrites dans
cubedefs.htm.

Toutes les 48 symétries sont uniquement générées par le produit

(S_URF3) x1 * (S_F2) x2 * (S_U4) x3 * (S_LR2) x4

avec x1 de 0 .. 2, x 2 de 0 .. 1, x3 de 0 .. 3 et x4 de 0 .. 1. Cette table (x1, x2, x3, x4) est composée
des nombres naturels de 0 .. 47 par 16 * x1 + 8 * x2 + 2 * x3 + x4

De cette façon, chacune de ces symétries est associé à un index de 0 .. 47. Avec S (i) qui dénote que la symétrie
qui appartient à l'indice i.

Deux cubes avec les permutations A et B sont équivalents si et seulement si il existe un i avec

S (i) -1 * A * S (i) = B

Tous les cubes qui sont équivalents, appartiennent à la même classe d'équivalence.
Dans le Cube Explorer S (i) est encodée dans les tableaux CornSym et EdgeSym de l'unité symmetries.pas

Vue mathématique des Coordinations: les Classes d'équivalences

(A mathematical view of Coordinates: Cosets)

Si vous avez un groupe G et un sous-groupe H, puis, pour chaque g de G l'ensemble (a * g de H) est une Table droite de H .

Chaque cube codé peut être considérée comme une permutation avec ses orientations attachées. Toutes ces permutations définissent le groupe G.
Toutes les valeurs de coordonnées utilisée dans Cube Explorer peuvent être mappé dans une Table d'équivalence droite, où le sous-groupe H mentionnés ci-dessus est définie par le type de la coordonnée.

Le tableau suivant décrit les sous-groupes H, qui correspondent aux différents types de coordonnées utilisées dans Cube Explorer.

Certaines de ces coordonnées sont réduites par les symétries "dans un deuxième temps, avant d'être utilisées.
Nous en discuterons dans le prochain chapitre.

Prenez par exemple le sous-groupe C0, qui définit les coordonnées d'orientation des coins

C0 = (g G avec de g (x). O = 0 pour tous les coins x)

Dans ce cas, les Tables sont définies par

subgroup of G	coset coordinate	used in
all permutations with corner orientations = 0	corner orientations coordinate range 0..2186	phase 1, optimal solvers
all permutations with edge orientations = 0	edge orientation coordinate range 0..2047	phase 1, optimal solvers
all permutations which leave the four UD-slice edges in their slice	UDSlice coordinate range 0..493	phase 1, optimal solvers
all permutations with edge orientations = 0 and which leave the four UD-slice edges in their slice.	FlipUDSlice coordinate range 0..494*2048 - 1	phase 1, standard optimal solver
all permutations which leave the corners in their place with arbitrary twist.	corner permutation coordinate range 0..40319	phase 2, optimal solvers
all permutations which leave the 8 edges of the U-face and D-face in their place with arbitrary flip	phase 2 edge permutation coordinate range 0..40319	phase 2
all permutations which leave the four UD-slice edges in their place with arbitrary flip	UDSliceSorted coordinate range 0..11879	phase 2, optimal solvers

A vos Dicos Anglais-Français pour ce tableau … facile !!

C0 = (g * a * g | a de C0)

Pour chaque élément d'un C0 et un élément g de G et de tout coin x, nous pouvous regarder la définition de la multiplication

(a * g) (x). o = a (g (x). c). o + g (x). o = 0 + g (x). o = g (x). o

Ainsi, tous les éléments de la Table droite C0 * g ont les mêmes orientations de coins (défini par la permutation g) et tous les
éléments de C0 * g ont les mêmes orientations de coins. Et si, d'autre part deux permutations
ont les mêmes orientations de coins, elles sont dans la même Table (omettons la preuve ici). Donc, il ya
une à une correspondance entre les coordonnées orientations de coins et les coordonnées des Tables définies par C0.

Coordonnées et Symétries

(Coordinates and Symmetry)

Les Coordonnées représentent Tables d'équivalences, chaque Table contient les nombreuses permutations.

Si vous déplacez le cube et le recolorez ou faites une conjugaison avec la symétrie S (i), comme expliqué Il ya des chapitres, les coordonnées changent généralement.

Vous devez être prudent si vous voulez créer une carte de coordonnées issue d'une conjugaison de symétrie dans une autre coordonnée.
Si vous avez deux permutations P et Q dans la Table S (i) -1 * P * S (i) et S (i) -1 * Q * S (i) doivent toujours être dans la même Table, sinon vous ne pouvez pas faire cette carte, et ne pouvez définir la Table équivalente.

Cela limite les symétries applicable ici. Il n'est pas difficile de montrer que ce sont exactement les symétries S (i) qui sont applicable, pour lesquelles le sous-groupe H définit la Table qui a la propriété S (i) 1 * H * S (i) = H.

coordinate	full symmetry group with 48 elements	subgroup generated by S_F2, S_U4 and S_LR2 with 16 elements	used in
corner orientation (twist)	no	yes	phase 1, optimal solvers
edge orientation (flip)	no*	no*	-------
UDSlice	no	yes	-------
FlipUDSlice	no	yes	phase 1, optimal solver, 64430 equivalence classes
corner permutation	yes**	yes	phase 2, 2768 equivalence classes
phase 2 edge permutation	no	yes	phase 2
UDSliceSorted coordinate	no	yes	huge optimal solver, 788 equivalence classes

Le tableau suivant montre, quelles est symétries sont applicable à quelles coordonne et où celui-ci est utilisé.

* Il est possible de donner une autre définition de la tranche d'orientations, de sorte que la totalité du groupe de symétrie peut être utilisé avec les coordonnées d'orientation de l'arrête . Mais nous préférons la définition usuelle qui est mieux adaptée à l'algorithme en deux phases.
** Non utilisé dans Cube Explorer

Pour les coordonnées de FlipUDSlice , les coordonnées de permutation de coins et les coordonnées UDSliceSorted de coordonner, le nombre de classes d'équivalence est donnée dans le tableau.

Prenez par exemple les coordonnées de theFlipUDSlice La portée de cette coordonner x est 0 .. 495 * 2048-1. Mais, par

conjugaison de symétrie, ces 1.013.760 coordonnées sont "réduite" de 64.430 classes d'équivalence.

Jusqu'à 16 de coordonnées appartenant à chaque classe d'équivalence -16 et pas 48, car nous utilisons uniquement des symétries qui préservent l'axe-UD. Pour chaque classe d'équivalence, nous stockons les plus petites coordonnées comme représentantes de cette classe dans un tableau d'une taille 64430. En général, nous appelons cela le tableau ClassIndexToRepresentantArray.

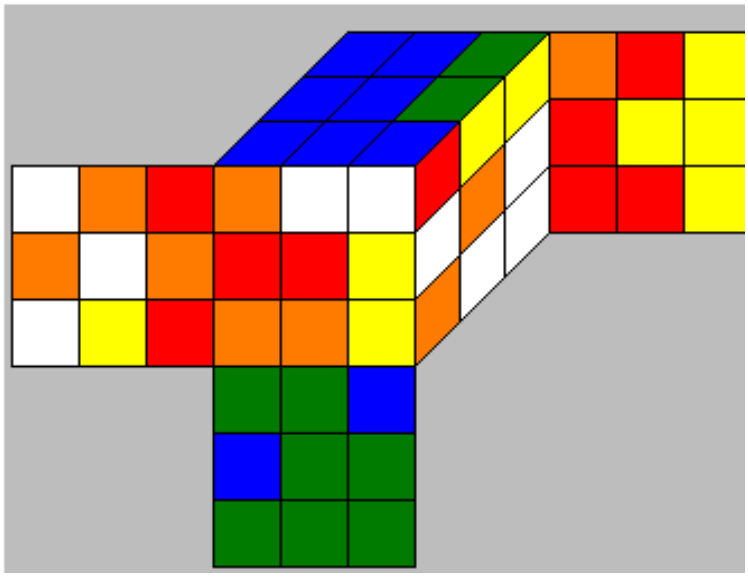
L'ancienne coordonnée de l'entier x est alors remplacé par une nouvelle coordonnée 16 * y + i, où y est l'index de la l'équivalence, il appartient à la classe (0 .. 64429) et i est l'index de symétrie (0 .. 15) qui transforme la de coordonnée représentant x. Pour distinguer les anciennes et les nouvelles coordonnées dans le texte, nous appelons les anciennes coordonnées une raw-coordonnée(coordonnée-racine) les nouvelles sym-coordonnées(?) dans le texte ci-après.

Ce symp-coordonnées a 64430 * 16 = 1.030.880 valeurs différentes, ce qui représente plus de 495 * 2048 = 1.013.760.
La raison en est que, pour certaines symétries de cubes , il y a de moins de 16 coordonnées originales pour une classe d'équivalence et le remplacement par les sym-coordonnées n'est pas unique, car 16 * y + i1 et 16 * y + i2 décrivant les mêmes raw-coordonnées pour quelques x i1 et i2 ..

Les coordonnées de l'Algorithme en deux Phases

(The Two-Phase Algorithm Coordonnées)

Si vous tournez les faces d'un cube résolu sans utiliser les mouvements R, R, L, L, F, F ', B et B', vous
créez seulement un sous-ensemble de tous les cubes. Ce sous-ensemble est désigné par G1 = <U,D,R2,L2,F2,B2>.



Une représentation typique de la G1 donne ce qui suit:

Dans ce sous-ensemble, l'orientation de tous les coins et tous les bords sont de 0. Et les quatre bords de la tranche-UD (entre la face-U et la face*D) restent isolés dans cette tranche.

D'autre part, si l'orientation de tous les coins et tous les bords est 0, et que les quatre bords de la tranche-UD sont dans leur tranche, nous avons un élément de G1.

L' Algorithme en Deux Phases(The Two-Phase algorithm) permet de résoudre le Cube par étapes.

Au cours de la phase 1, l'algorithme cherche des manœuvres qui vont transformer un cube brouillés en G1. Ainsi, l'orientation des coins et des bords doit être limitée et les bords de la tranche*UD doivent être transférées dans cette tranche. Au cours de la phase 2, nous rétablirons le cube.

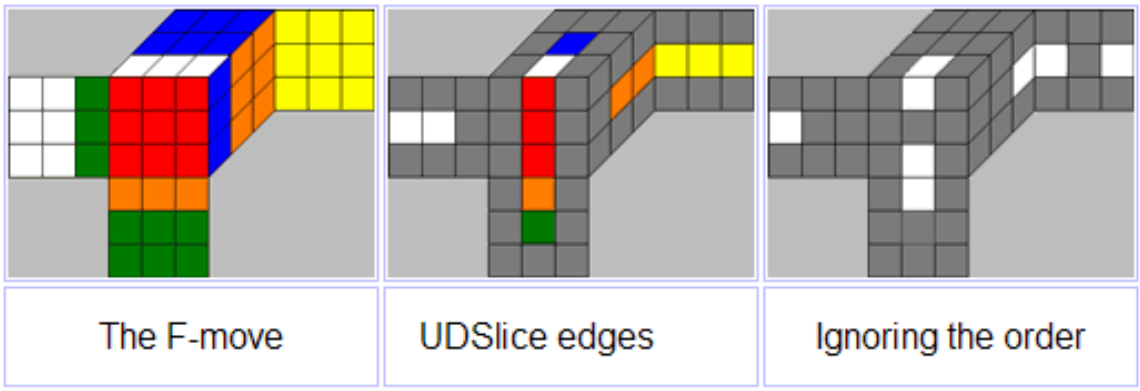
Il existe de nombreuses possibilités pour les manœuvres dans la phase 1. L'algorithme essaie différentes phase 1
manœuvres de trouver la solution d'ensemble la plus courte possible.

Au cours de la phase 1, tout le cube est décrit par trois coordonnées:

les coordonnées d'orientation des coins(0 .. 2186),les coordonnées d'orientation des bord (0 .. 2047) et les coordonnées UDSlice

les coordonnées UDSlice forment un nombre compris entre 0 à 494 (12 * 11 * 10 * 9 / 4! -1) Qui est déterminé par la position des 4 bords UDSlice . L'ordre des 4 bords UDSlice à l'intérieur de la position est ignoré.

Nous prenons le mouvement-F (F-move)à titre d'exemple:



Le mouvement-F UDSlice Arrêtes Ordre ingnoré

La fonction suivante à partir de cubicube.pas met en œuvre le calcul de cette coordination. L'explication de ce fonctionnement n'est pas évidente et elle est expliquée plus en détail ici. C (n, k) est le coefficient binomial (n choisit k).

```
function CubieCube.UDSliceCoord;
var s: Word; k,n: Integer; occupied: array[0..11] of boolean; ed: Edge;
begin
  for n:= 0 to 11 do occupied[n]:=false;
  for ed:=UR to BR do if PEdge^[ed].e >= FR then occupied[Word(ed)]:=true;
  s:=0; k:=3; n:=11;
  while k>= 0 do
  begin
    if occupied[n] then Dec(k)
    else s:= s + C(n,k);
    Dec(n);
    end;
  Result:= s;
end;
```

Ainsi, chaque cube relevant de la phase 1 est décrit par une coordonnée triple (x1, x2, x3), et le triplet est (0,0,0) si et seulement si nous avons un cube de G1. L'espace du problème de la phase 1 a 2187 * 2048 * 495 = 2.217.093.120 différents Etats.

Au cours de la phase 2, tout le cube est aussi décrit par trois coordonnées:

Les Coordonnées de permutation des coins (0 .. 40319), les Coordonnées de permutation des arrêtes en phase 2 (0 .. 40319), et les Coordonnées de permutation de UDSlice en phase 2 (0 .. 23).

Le triplet (0,0,0) de la phase 2 appartient à un cube parfait.

les Coordonnées de permutation des arrêtes en phase 2 sont semblables à la description des niveaux coordonnées .
elle n'est valable que dans la phase 2.

Nous avons 8! = 40320 possibilités de permuter les 8 bords des faces U et D (n'oubliez pas que nous ne laissons Tourner à 180 degrés que les faces R, L, F et B).

```
function CubieCube.Phase2EdgePermCoord: Word;
var i,j: Edge; x,s: Integer;
begin
```

```
  x:= 0;
  for i:= DB downto Succ(UR) do
  begin
    s:=0;
    for j:= Pred(i) downto UR do
    begin
      if PEdge^[j].e>PEdge^[i].e then Inc(s);
    end;
      x:= (x+s)*Ord(i);
  end;
  Result:=x;
end;
```

Les coordonnées UDSlice de la phase 2 devrait avoir un domaine de 0 .. 23, car elles représentent 4! permutations de bords de UDSlice dans leur tranche. Mais nous utilisons une extension des coordonnées de UDSlice, qui est utilisée dans le grand solutionneur optimalde toute façon où nous vérifions également l'ordre des quatre arrêtes. Ce "tri" de coordonnées a un domaine 0 à 11879 = 12 * 11 * 10 * 9.1. Mais, dans la phase 2 ces coordonnées prennent une valeur comprise entre 0 et 23.

Voici l'implémentation dans cubicube.pas:

```
function CubieCube.UDSliceSortedCoord: Word;

var j,k,s,x: Integer; i,e: Edge; arr: array[0..3] of Edge;
begin
   j:=0;
  for i:= UR to BR do
  begin
    e:=PEdge^[i].e;
    if (e=FR) or (e=FL) or (e=BL) or (e=BR) then begin arr[j]:= e; Inc(j); end;
  end;

  x:= 0;
  for j:= 3 downto 1 do
  begin
    s:=0;
    for k:= j-1 downto 0 do
    begin
      if arr[k]>arr[j] then Inc(s);
    end;
    x:= (x+s)*j;
  end;
  Result:= UDSliceCoord*24 + x;
end;
```

Le problème du domaine de la phase 2 a
40320 * 40320 * 24 / 2 = 19.508.428.800
différents Etats.

Les Tables de Permutations)

(The Move Tables)

Si vous appliquez l'une des 18 rotations de faces possibles (a "move") du cube, la permutation des coins et le changement des bords . dans le niveau des coordonnées, les tableaux se mouvent de coordonnées à coordonnées.

Cette cartographie est possible, car nous pouvons montrer que si on applique un mouvement M sur deux permutations a et b avec la même coordonnée x, les deux résultats ont les mêmes coordonnée x '. Si A et B ont la même coordonnée x, et H est le sous-groupe pour la définition de Coordonnées de cette Table d'équivalences, il existe une permutation g du groupe G du cube aussi a et b sont des éléments de H * g (rappelez-vous que nous utilisons des Tables Droites). Alors a*M,
etb* M sont bien sûr des éléments de [H * g] * M = H * [g * M] et, partant, sont dans la même Table et ont les mêmes coordonnées x '.

les Tables de permutations sont des tableaux à deux dimensions qui décrivent la façon dont cette transformation est faite.
Nous faisons la distinction entre permuter les tables pour les "simples" raw-coordonnées et permuter les tables0 des sym-coordonnées, qui sont réduites par symétries.

Permuter les tables pour les coordonnées-racines
(Move tables for raw-coordinates)

Toutes les tables de raw-coordonnées ont la même structure. Prenons par exemple la permutation de la table de coordonnées d'orientation des arrêtes:

```
TwistMove: array [0 .. 2187-1, UX1 .. Bx3] of Word;
```

Si vous demandez par exemple le mouvement R2, TwistMove [oldCoordinate, RX2] donne les nouvelles coordonnées. C'est fait assez rapidement par rapport à une permutation en niveau cubie ou en niveau facette.

Voici la documentation du code de CordCube.pas pour générer le mouvement de tableau:

```
procedure CreateTwistMoveTable;
var c: CubieCube; i,k: Integer; j: TurnAxis;
begin
  c:= CubieCube.Create;//create a cube c on the cubie level
  for i:=0 to 2187-1 do
  begin
    c.InvCornOriCoord(i);//generate a permutation with corner orientation i
    for j:= U to B do
    begin
      for k:= 0 to 3 do //k=3 restores the original state
      begin
        c.Move(j);//apply all 18 face turns on c
        if k<>3 then TwistMove[i,Move(3*Ord(j)+k)]:=c.CornOriCoord;//save result
in the array
      end;
    end;
  end;
  c.Free;
end;
```

Permuter les tables de coordonnées-Symétriques
(Move tables for sym-coordinates)

Si nous réduisons les coordonnées par symétries, nous ne faisons générer un mouvement que pour la table des représentant la classes d'équivalence. R(j) est une permutation appartenant à la représentation d'équivalence de la classe indice j.

Lorsque l'on applique un mouvement M sur cette représentation, le résultat sera dans une autre équivalence de la classe k, de sorte qu'il y a une symétrie S(i) avec R (j) * M = S (i) -1 * R (k) * S (i). il en résulte une entrée de Table permutable correspondant aux sym-coordonnées,c'est MoveTable [j,M]: = 16 * k +i.

Voici un exemple pour la permutation de table de FlipUDSlice dans cordcube.pas (toutes les parties sans importance ôtées):

```
procedure CreateFlipUDSliceMoveTable;
var c: CubieCube; i,k,n: Integer; j: TurnAxis;
begin
  SetLength(FlipSliceMove,64430,18); //18 different faceturns
  c:= CubieCube.Create;
  for i:=0 to 64430-1 do //iterate over all equivalence classes
  begin
    n:= FlipUDSliceToRawFlipUDSlice[i]; //get the raw-coordinate of the
representant
    c.InvUDSliceCoord(n div 2048); //and generate a permutation which has this
FlipUDslice coordinate
    c.InvEdgeOriCoord(n mod 2048);
    for j:= U to B do
    begin
      for k:= 0 to 3 do
      begin
        c.Move(j); //apply all 18 faceturns
        if k<>3 then FlipSliceMove[i,3*Ord(j)+k]:= c.FlipUDSliceCoord; //the
sym-coordinate
      end;
    end;
  end;
end
```

La procédure pour trouver les sym-coordonnées donnant une permutation P n'est pas vraiment difficile, mais un peu plus compliqué que le calcul des raw-coordonnées. Nous avons seulement besoin de cette procédure dans la phase d'initialisation où nous devons calculer les coordonnées du cube que nous voulons résoudre.

Pour 0 <= i <16, nous appliquons S (i) * P * S (i) -1 et calculons les raw-coordonnées jusqu'à que l'on trouve les raw-coordonnées dans ClassIndexToRepresentantArray à une position k. Laissez-nous désigner cette coordonnée par R(k).

S (i) * P * S (i) -1 = R (k) est équivalent à S (i) -1 * R (k) * S (i) = P, cela signifie P a pour sym-coordonnée 16 * k + i.

Regardez la fonction CubieCube.FlipUDSliceCoord dans cubicube.pas pour un exemple.

Appliquer un mouvement est aussi plus compliqué pour les sym-coordonnées par rapport aux raw-coordonnées, car nous avons seulement construit une table de mouvements pour les représentants des classes d'équivalence. Mais l'avantage de l'utilisation des sym-coordinées est de réduire les grandes tables par un facteur d'environ 16, c'est beaucoup plus valable que le désavantage causé par l'augmentation de la complexité.

Si nous avons les sym-coordonnées x, nous pouvons extraire de ces coordonnées l'indice j de la classe d'équivalence et l'indice i de la symétrie. Pour un déplacement M, nous avons l'associativité de la permutation de groupe et dénotons le représentant de la classe d'équivalence R (j):

[S (i) -1 * R (j) * S (i)] * M = [S (i) -1 * R (j) * S (i)] * M * [S (i) -1 * S (i)] = [S (i) -1 * R (j)] * [S (i) * M * S (i) -1] * S (i)

[S (i) * M * S (i) -1] est la conjugaison d'un passage par une symétrie qui est un autre mouvement. Dans le symmetry.pas SymMove array [SymIdx, Move] est initialisé, ce qui SymMove [i, M] donne le résultat souhaité. Laissez-nous désigner le résultat en M1.

Nous devons donc à calculer

[S (i) -1 * R (j)] * M1 * S (i) = S (i) -1 * [R (j) * M1] * S (i)

La sym-coordonnée y pour [R (j) * M1] peut être lue sur MoveTable [j, M1]. De y extrayons les indice de classe j1 et l'index de symétrie i1. Cela signifie que [R (j) * M1] = S (i1) -1 * R (j1) * S (i1).

Nous avons donc

S (i) -1 * [R (j) * M1] * S (i) = S (i) -1 * [S (i1) -1 * R (j1) * S (i1)] * S (i) = [S (i) -1 * S (i1) -1] * R (j1) * [S (i1) * S (i)]

et parce que [S (i) -1 * S (i1) -1] = [S (i1) * S (i) -1] on peut écrire que

[S (i1) * S (i) -1] * R (j1) * [S (i1 * S (i)].

[S (i1) * S (i)] est le produit de deux symétries, donc une autre symétrie S (i2). Le tableau SymMult [SymIdx, SymIdx] créé en symmetries.pas ne fait pas ce calcul. Laissez-nous désigner SymMult [i1, i] avec i2.
Ainsi, notre résultat est

S (i2) -1 * R (j1) * S (i2) et de la sym-coordonnées correspondante est de 16 * j1 + i2.

Ainsi,comparé avec les tables de mouvements raw-coordonnées où nous avons seulement besoin d'une table-verrouillée nous besoin de trois table-verrouillées dans les tableaux SymMove , MoveTable et SymMult.

Les Tables élaguées

(Pruning Tables)

La vitesse de l'Algorithme en Deux Phases et du Solvers Optimal dépends de leur capacité à donner la Limite la plus faible pour le nombre de coups nécessaire à ramener le cube à son état initial, car il permet à l'élagage des arbres au cours de la recherche.
L'objectif est un cube résolu grâce au Solvers Optimal et à la deuxième phase de l'Algorithme en Deux Phases associé à un certain sous-groupe G1 dans le cas de la première phase de l'Algorithme en Deux Phases.

Nous basons l'élagage des tables dans les coordonnées. Rappelez-vous que des coordonnées ou une table de plusieurs coordonnées (Tuple) représentent une table d'équivalence pour un sous-groupe H (si on utilise une table de coordonnées, le sous-groupe H est l'intersection des sous-groupes définissant une coordonnées seule). Il se coordonne lui-même ou un indice calculé à partir de deux ou trois coordonnées de définit la position dans cette table élaguée. A cette position, nous stockons le nombre de coups qui sont nécessaires pour ramener le cube au sous-groupe H.
Parce que cet objectif est toujours inclus dans H (phase 2 du solveur optimal) ou est identique à H (phase 1), le nombre de mouvements enregistrés dans les table élaguées est toujours le plus petit nombre de coups permettant de remettre le
cube à l'État initial. C'est essentiel pour permettre à l'algorithme de travailler.

Nous avons besoin des tables élaguées pour la phase 1 et phase 2 de l'algorithme en deux phases et pour le solver optimal en mode large . Les tables élaguées du solveur optimal en mode standard sont identiques tables élaguées de la phase1.

Dans les trois cas, la position des tables élaguées est calculée à partir d'une sym-coordonnée et d'une ou deux raw- coordonnée.

Pruning Table	Sym-coordinate	Raw-coordinate(s)	Number of Table-Entries	Maximal pruning depth
Phase 1	FlipUDSlice (40320 equivalence classes)	Corner Twist x = UDTwist (2187 cases)	88,179,840	12
Phase 2	Corner Permutation (2768 equivalence cases)	Edge Permutation x (40320 cases)	111,605,760	18
Huge Optimal Solver	UDSliceSorted (788 equivalence cases)	Edge Flip x1 (2048 cases) Corner Twist x2 (2187 cases)	3,529,433,088	13

Si vous êtes intéressé par la distribution exacte des valeurs de Tables élaguées jetez un oeil à ce tableau

Les FlipUDSlice sym-coordonnées, de cet exemple à la paire (y, i), où y est l'index de la classe d'équivalence et i est l'indice correspondant à la symétrie,x est la rotation de coin.
P réalise la permutation du cube appartenant à ces indices..
Puis, par la conjugaison de S (i) * P * S (i) -1, nous avons un cube qui à même distance du cube fini,de la sym-coordonnée FlipUDSlice sym-coordonnées (y, 0) et de la rotation de coin x '. Ensuite, la position dans la table élaguée est calculée par 2187 * y + x ».

Le principe suivant vaut pour le calcul des indices dans toutes les tables élaguées:
Extraction de l'indice i (0 <= i <16) de symétrie des coordonnées symboliques et transformer le cube en conjugaison avec S (i)en un cube équivalent ayant le même indice de classe d'équivalence y, mais une symétrie avec indice 0.
Transformer la raw-coordonnée x (ou x1 et x2 dans le cas du Solver Optimal en mode large) en x '(x1, x2).

L'indice de la table élaguée au cours de la phase 1 est alors calculé par 2187 * y + x', en phase 2 par 2768 * x' + y (hmmm, pourquoi n'ai-je pas pris 40320 * y + x '?) et dans le solveur mode large par (2048 * y + x1') * 2187 + x2 '.

La transformation des indices racine (raw indices) est faite des tables (voir le code source, CordCube.pas)

```
TwistConjugate: array[0..2187-1,0..15] of Word;
FlipConjugate: array of array of array {[0..2048-1,0..15,0..788-1]}of Word;
Edge8PosConjugate: array[0..40320-1,0..15] of Word;
```

Vous pouvez voir que la conjugaison de permutation des bords qui est utilisée dans les tables élaguées du solveur en mode large est un peu plus compliquée. La raison en est que, comme indiqué ici le sous-groupe qui définit les coordonnées de permutation des Tables d'équivalences n'est pas compatible avec les 16 symétries. Mais si vous ajoutez les informations de la classe d'équivalence UDSliceSorted
classe, la transformation est possible.

Pour réduire la taille de la mémoire, nous ne gardons pas le nombre de coups, mais seulement le nombre de coups modulo 3. Cela est possible parce que chaque rotation de face (faceturn)change le nombre de coups que par -1, 0 ou 1. Donc, si vous appliquer un faceturn vous êtes en mesure de suivre le nombre de coups, et si vous connaissez le nombre de coups de refaire le cube initial.

Ce nombre pour l'état initial peut être reconstruit avec la table mod 3: De l'état initial essayez,lequel des 18 faceturns est diminué modulo 3 (il doit y avoir un faceturn avec cette propriété, ou vous êtes retourné au cube initial). Répétez cette opération jusqu'à ce que vous ayez atteint l'objectif de l'État initial et comptez le nombre de coups nécessaires.(procédures GetPrun, GetPrunBig
et fonction GetPrunPhase2 dans CordCube.pas)

Au cours de la génération de table, nous utilisons 2 bits pour chaque entrée, car nous avons besoin d'un quatrième état de marquer une entrée vide.
Ensuite, nous compressons le tableau de stockage, 5 entrées par octet, en utilisant seulement les bits 1,6 pour chaque entrée.
Nous ne le faisons pas en linéaire, comme la compression (0,1,2,3,4), (5,6,7,8,9), (10,11,12,13,14) ... mais de cette manière (0 , 1,2,3, x), (4,5,6,7, x +1), (8,9,10,11, 2),.... x où x est d'environ 4 / 5 du nombre total des entrées. grâce à cette façon,pas besoin de (div 5) et (mod 5), et un calcul arithmétique beaucoup plus rapide avec (div 4) et (mod 4).

Nous générons la table, dans le mode première "recherche-avant"(breadth-first "forward-search") Nous stockons la profondeur 0 à la position cube initial et l'appliquons l'ensemble des 18 mouvement à cette position . Àux positions correspondantes, nous sauvons la profondeur 1.
Dans le passage suivant,nous appliquons les 18 états correspondant à ces positions dans la Table élaguée taille qui a une entrée 1 .

Nous écrivons 2 a cette position résultante si elle est marquée comme vide etc ..

Si vous jetez un coup d'œil par exemple à CreateFlipUDSlicePruningTable dans CordCube.pas vous pouvez le voir, que le code n'est pas aussi simple que décrit ci-dessus. Parce que nous utilisons une sym-coordonnées (FlipUDSlice) avec une raw-coordonnée (UDTwist) pour calculer l'index dans la table élaguée, nous construirons une mauvaise table si nous ne procédons pas avec beaucoup de soin.

Le problème est la permutations A, où la symb-coordonnée n'est pas unique, car la permutation a elle-même quelques symétries.
en effet (y, i1) et (y, i2) correspondent à deux classindex-symmetryindex paires des Coordonnées FlipUDSlice qui appartiennent à la même raw-coordonnée. Laissez les UD-Twist coordonner la permutation de A en x. L'indice de la table élaguée est calculé par 2187 * y + x ', où x' est la coordonnée UDTwist des permutations S (i1) * A * S (i1) -1 et S (i2) * A * S (i2) -1. Parce que ces 2 permutations ont généralement différentes coordonnées UDTwist, il y a plus d'une position dans la table élaguée, nous l'avons la comblé .
Il nous faut donc analyser avec soin les symétries de la coordonnée FlipUDSlice.

Si il n'y a pas beaucoup de vide à gauche des entrées dans la table élaguée, nous basculons vers "recherche arrière"("backward search).
Nous appliquons les 18 mouvements à toutes les permutations vides qui appartiennent à ces entrées et regardons si le résultat est une permutation qui a une entrée correspondant à la profondeur d de la dernière passe. Dans ce cas, nous remplissons l'entrée avec d +1.
De cette manière, nous sauvons un temps considérable lors de la génération des tables.

L'Algorithme à Deux Phase dans le détail

(Two-Phase Algorithm Details)

J'ai développé l'algorithme à deux phases en 1991 et 1992. Il est inspiré de l'algorithme de la Thistlethwaite
pour résoudre le cube. Sa méthode consiste à travailler par le biais de la séquence de sous-groupes suivante:
H0 = <L,R,F,B,U2,D2>, <L,R,F2,B2U2,D2> = H1, H2 = <L2,R2,F2,B2,U2,D2> de trouver une solution.
Il a utilisé des tableaux statiques pour les manœuvres et l'algorithme a besoin d'au plus 52 coups.

Réduire le nombre de sous-groupes intermédiaires donnerait de de plus courtes solutions, et j'ai décidé d'utiliser un seul sous-groupe G1 = <U,D,R2,L2,F2,B2> qui est équivalent à H1 de Thistlethwaite. Mais il est clair que, dans ce cas des tableaux statiques pour les manœuvres sont impossibles en raison de la taille du sous-groupe. Donc, ces manœuvres devait être calculé dynamiquement au cours de la résolution de procédure. Avec mon matériel, j'ai utilisé (Atari ST 8 MHZ avec 1 Mo de RAM), cela était loin d'être trivial, car il y a environ 2217 millions de positions différentes dans phase 1 (dans G1) et environ 19,508 millions de postes dans la phase 2 (demandées pour résoudre le cube dans G1).

Après une longue lutte, j'ai finalement trouvé les ingrédients du travail de recherche de manœuvres:

- Cartographie des permutations et des orientations pour les nombres naturels et la mise en œuvre de tables verrouillées (lookups) pour ces chiffres.
- Calculer avec ces chiffres des indices pour les tableaux qui contiennent des informations relatives à la distance de la solution finale.

La Phase 1 a besoin d'au plus 12 coups (voir la distribution) et la phase 2 a besoin d'au plus 18 coups (Michael Reid démontra cela en 1995, ne voie pas la distribution parce que les Tables élaguées de la phase 2 ne contiennent que 1 / 24 de tous les positions possibles de cette phase 2). Ainsi, la première solution générée par l'algorithme à deux phases sera toujours au plus 30 mouvements.
L'idée de combiner les solutions optimale de la phase 1 avec des solutions optimales de la phase 2 pour obtenir une solution d'ensemble plus courte est tout à fait évidente, donc, mais j'ai été surpris de voir comment la longueur de l'ensemble des solutions habituellement 22 déplacement par secondes ou moins sur l'Atari ST et de 20 déplacements/Sec ou moins dans la mise en œuvre actuelle d'un PC 2000.

Je n'ai pas utilisé la réduction de symétrie dans cette première version de l'algorithme en deux phases. L'idée de réduction de symétrie nait de Mike Reid qui l'a utilisé en 1997 pour tenir une table de la phase 1 dans la mémoire vive (RAM) puis,dans la phase de son solveur optimal.

Depuis Cube Explorer 2 de réduction de la symétrie est aussi utilisé.

Au cours de la phase 1, nous utilisons deux coordonnées: La coordonnée FlipUDSlice (un sym-coordonnée avec 64430 différentes classes, combinant les coordonnées d'orientation des arrêtes et la coordonnée UDSlice) et les coordonnées d'orientation des coins.

Dans le calcul de l'indice des Tables élaguées, les deux coordonnées sont utilisées. Cela signifie, que nous avons une entrée pour chaque position de la phase 1.

Au cours de la phase 2, nous avons le problème d'initialisation des trois phases de 2 de coordonnées, permutation des coins, phase 2 UDSlice et la phase 2 permutation des arrêtes.

Parce que la phase 2 UDSlice etla phase 2 des coordonnées de permutation des arrêtes ne sont pas définies dans la phase 1, nous devrait revenir au niveau cubie appliquer notre solution phase 1 avant de lancer le calcul de coordonnées Phase 2 du cube.

Nous avons donc recours à trois coordonnées d'aide qui sont également définis dans la phase 1 (UDSliceSorted, RLSliceSorted et FBSliceSorted), chacune décrivant la position exacte des 4 arrêtes (bords) d'une tranche.
Helper-coordonate div-24 donne la position , Helper-coordonate mod-24 décrit les permutations possibles des 4 bords à l'intérieur de cette position.

La phase 2 UDSlice coordinate est identique à la UDSliceSorted coordinate au cours de la phase 2, de manière nous ne devons pas refaire tous les calculs.

La phase 2 2edge permutation coordinate peut être extraite de la RLSliceSorted et et FBSliceSorted coordinate avec l'aide de la table GetEdge8Perm d'une taille de 11880 * 24. GetEdge8Perm [RLSliceSorted, FBSliceSorted mod 24] qui donne les coordonnées. Nous pouvons aussi utiliser FBSliceSorted mod 24, parce que la position de la partie information de la position des tranches des cubies FB redondante.

Le corner permutation coordinate est déjà définit dans la phase 1. Nous utilisons les raw-coordonnées (0 .. 40329) dans la table de mouvement (movetable). Avant la construction de l'index dans la table élaguée (avec la phase 2 de coordonnées de permutation des arrêtes), il est associé à un sym-coordinate par 2768 classes.

Comme déjà mentionné ci-dessus, l'algorithme ne s'arrête pas lorsque la première solution est trouvée, mais continue à recherche de solutions plus courtes en intégrant à la phase 2 des solutions sous-optimales de la phase 1.

Par exemple, si la première solution est de 10 déplacements en phase 1 suivi de 12 mouvements en phase 2, la deuxième solution peut avoir 11 déplacements en phase 1 et seulement 5 coups en phase 2. La longueur de manœuvres de la phase 1 augmente et la longueur de manœuvres de la phase 2 diminue.

Habituellement, la longueur la phase 2 baisse très rapidement (généralement inférieure à 9). La performance de l'algorithme augmente considérablement si nous n'initialisons pas les trois coordonnées lors de l'entrée dans la phase 2, mais seulement les coordonnées de permutation des coins.
Une petite Table élaguée seulement pour ces coordonnées est vue dans plusieurs cas , cela même si les coordonnées de permutation des coins ne peuvent pas être restauré au sein de ce petit nombre de coups.
Ainsi, nous pouvons revenir immédiatement à trouver le prochaine sous-solution optimale en phase 1.

Une autre façon d'augmenter considérablement le rendement est de supprimer jeter certaines sous-solution optimale en phase 1.
Si la manœuvre M définit une solution de la phase 1, puis, par exemple M R2 ou MU F2 sont bien sur aussi des sous-solution optimale en phase 1 , parce que R2, F2 et U sont des déplacements de la phase 2. Mais ces solutions ne sont pas pertinentes, parce que la phase 2 appliquée à M R2 ne sera jamais une solution globale plus courte que la phase 2 appliquée à M.

Au cours de l'actuelle mise en œuvre, nous jetons toute sous-solution de manœuvre optimale de la phase 1, si certaines sous-manœuvres commençant par le premier pas sont déjà une solution de la phase 1.
Nous pourrions perdre des solutions avec cette façon qui n'est pas pertinente, sauf que, cet algorithme n'est pas fait pour des manœuvre optimale.
Cela se fait de mieux en utilisant le solutionneur optimal c'est sur !.

Prenez par exemple le cube C généré par RL U2 RL. F (6 coups). L'algorithme ne trouve pas la Solution F ». L 'R' U2 L 'R', car l'application de F 'à C amène le cube dans le sous-groupe G1 et est donc une solution phase 1. Toutes optimales solutions de la phase 1 commençant par F ' est rejetées.

Le Solveur ou Solutionneur Optimal (The Optimal Solvers)

Un solveur optimal n'a jamais besoin de plus mouvements pour rétablir un cube brouillé que le nombre de coups utilisés pour brouiller le cube.

Le solver optimal mode standard mis en œuvre dans Cube Explorer utilise la méthode de Mike Reid de 1997.
Nous faisons une triple phase 1 de recherche en parallèle dans trois directions différentes. Cela signifie que notre objectif, est l'intersection des groupes <U,D,R2,L2,F2,B2>, <U2,D2,R,L,F2,B2> et <U2,D2,R2,L2,F,B>. D'ailleurs,cette intersection n'est pas le groupe <U2,D2,R2,L2,F2,B2> mais un groupe de six fois plus grand.

Parce que dans la phase 1 la Table élaguée a une entrée pour chaque position possible, les solutions de la phase 1 sont générées très rapidement.
Alors nous produisons une triple phase 1 de solutions sous-optimales et les jetons-les jusqu'à ce que le cube soit résolu (le cube résolu est une solution de la phase 1).

Utiliser de la Table élaguée en parallèle dans trois directions différentes est une bonne chose car elle améliore sensiblement la qualité de l'élagage des arbres. Si P1, P2 et P3 sont des valeurs élaguées dans les trois directions différentes, nous pouvons utiliser max (P1, P2, P3) en tant que valeur effective de valeurs élaguées dans notre recherche.

Un regard sur la distribution de valeurs élaguées de la phase 1 montre que la probabilité d'avoir une valeur élaguée de 10 dans chaque direction est relativement élevé. L'idée suivante (suggéré par Michiel de Bondt) améliore la performance de l'algorithme d'environ 35%:
Si nous appliquons un mouvement arbitraire à un cube à partir de l'état final, la résultante du cube, sera au moins dans l'un des trois sous-groupes mentionnés ci-dessus. Cela implique que, au moins une des trois valeurs élaguées reste à 0. Donc, si nous faisons par exemple 10 coups depuis l'état résolu, au moins une des valeurs des valeurs élaguée est de 9 ou moins. Si une autre part des trois valeurs élaguées est de 10, nous savons que nous pouvons utiliser 11 comme valeur de l'efficacité de l'élagage.

En général:
si les trois valeurs élaguées sont n, on peut utiliser n +1 comme valeur élaguée effective.
Mon solveur optimal mode Large fonctionne de la même manière que le solveur optimal mode Standard. La seule différence est qu'il utilise les coordonnées UDSliceSorted au lieu des coordonnées UDSlice pour construire la Table élaguée. Ainsi, ce tableau est environ 24 fois plus grand et la valeur élaguée moyenne est plus élevée,
comme dans la distribution des Tables élaguées. il tourne environ 5 fois plus rapidement que le Solveur Optimal en mode Standard.

En 2005, j'ai développé un autre grand solutionneur optimal qui nécessite environ 3 Go de RAM. Il utilise les coordonnées du Solveur Optimal mode Standard et des coordonnées additionnelles, qui décrivent la position de quatre des huit coins du cube (70 possibilités). Ainsi, la taille de sa Table élaguée est 70 fois plus grandes que les tables du Solveur Optimal Standard.
il tourne environ 15 fois plus vite que le Solveur Optimal Standard.

Vous aimez Cube Explorer?

(Do you like Cube Explorer?)

Cube Explorer est un logiciel gratuit. ça a été amusant de développer l'algorithme et de le voir fonctionner correctement,
mais étendre l'idée vers un programme convivial a été le travail.
Si vous aimez Cube Explorer et que vous l'installez sur votre système, vous pouvez montrer votre appréciation pour ce travail
dans la forme que vous voulez. Il suffit de vous demander: Quelle valeur a ce programme pour moi?
Vous pouvez envoyer une jolie carte postale de l'endroit où vous habitez, un puzzle ou un livre intéressant,
de l'argent (voir Lien Paypal sur la page d'accueil) ou tout ce que vous voulez à mon adresse Physique:

Herbert Kociemba
Bismarckstr. 31
D-64293 Darmstadt
Germany

Toutes les suggestions pour des améliorations doivent être envoyés par Email à:
mailto:kociemba@t-online.de

La dernière version de Cube Explorer sera disponible à:

http://kociemba.org/cube.htm

Traduit vers le Français le 9/07/2009.