Commutator Subgroup

Links

Commutator Subgroup – Useful Links
Wikipedia – Commutator Subgroup
http://en.wikipedia.org/wiki/Commutator_subgroup
Wikipedia – Alternating Group
http://en.wikipedia.org/wiki/Alternating_group
G. A. Miller
http://www.ams.org/journals/bull/1899-06-03/S0002-9904-1899-00683-9/S0002-9904-1899-00683-9.pdf
E. Bertram
http://www.sciencedirect.com/science/article/pii/0097316572901021
L. Kappe, R. Morse
http://faculty.evansville.edu/rm43/publications/commutatorsurvey.pdf
H. Cejtin, I. Rivin
http://arxiv.org/PS_cache/math/pdf/0303/0303036v2.pdf

Basics

In abstract algebra, the commutator subgroup or *derived* subgroup of a group is the subgroup generated by all the commutators of the group.

For elements g and h of a group G, one of the two expressions of the commutator of g and h is $[g,h]: = ghg'h'$. The commutator $[g,h]$ is equal to the identity element e if and only if $gh = hg$, that is, if and only if g and h commute. In general, $gh = [g,h]hg$.

However, the product of two or more commutators need not generally be a commutator.

An alternating group is the group of even permutations of a finite set. The alternating group on the set $\{1,...,n\}$ is called the alternating group of degree n, or the alternating group on n letters and denoted by A_n.

A Property of Alternating Groups

It has been proved by G. A. Miller that the alternating group on n letters, $n >= 5$, consists *entirely* of commutators. This was rediscovered over half a century later by O. Ore.

They demonstrated that *any* permutation σ of A_n can be written as the product of 2 n-cycles ρ1, ρ2:

$$\sigma = \rho 1 \cdot \rho 2$$

Because permutation ρ2 is an n-cycle, as ρ1 and ρ1' are, and because permutations which show the same cycle structure are conjugates, there exist a permutation τ such that permutation ρ2 is the conjugate of ρ1' by τ:

$$\rho 2 = \tau \cdot \rho 1' \cdot \tau'$$

So that permutation σ can now be written as a commutator:

$$\sigma = \rho 1 \cdot \tau \cdot \rho 1' \cdot \tau' = [\rho 1, \tau]$$

E. Bertram showed later that permutations ρ1, ρ2 need not be n-cycles, but l-cycles, where the necessary and sufficient condition on l is:

$$(3n/4) \leq l \leq n$$

Commutator Example

The 24 facelets of a given orbit of corner-centers of a 7x7x7 cube can be uniquely identified from a set of 24 letters A…X. We can then define alternating group A_{24} as the group of all even permutations of these letters.

Applied to $n = 24$, the general condition on l gives:

$$18 \leq l \leq 24$$

We choose $l = 23$, ρ1 = [NR NL, NU R NB ND']* and τ = (R NF' L' NF R')*, so that both ρ1 and ρ2 are 23-cycles.

Notice that algorithm [NR NL, NU R NB ND'] is a 'pure' 23-cycle, ie. it will not move or rotate any facelet that doesn't belong to the selected orbit of corner-centers.

By using CubeTwister, it can be shown that the composition of ρ1 and ρ2 gives a 5-cycle, which can be written either as the product:

σ = ρ1·ρ2 = [NR NL, NU R NB ND'] (R NF' L' NF R') [NR NL, NU R NB ND']' (R NF' L' NF R')' (34 moves)

or as the commutator:

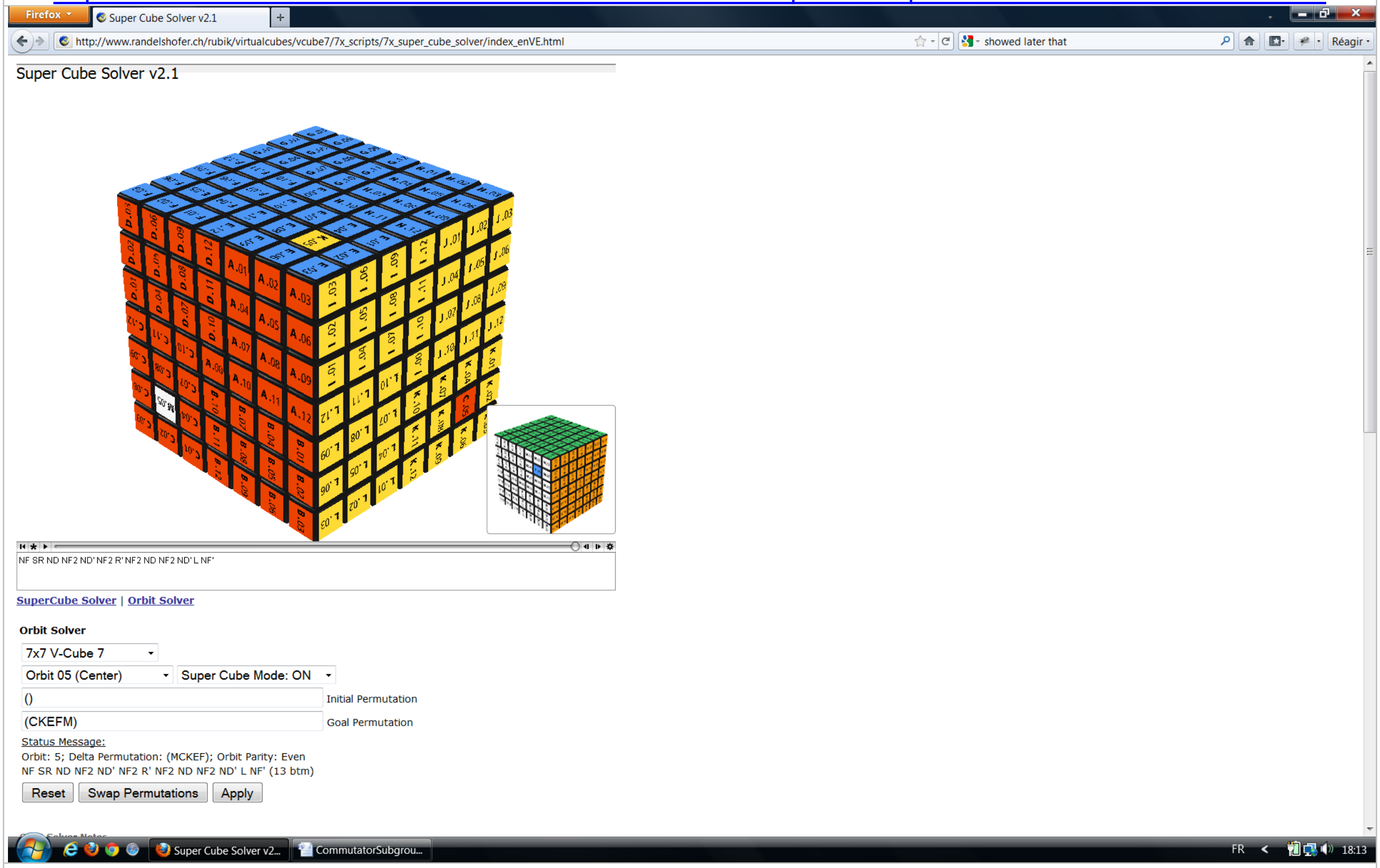
σ = [ρ1, τ] = [[NR NL, NU R NB ND'], (R NF' L' NF R')] (34 moves)

Factorizing permutations into 2 cycles of length l is generally not very efficient in terms of moves, though. We can search for a shorter algorithm using Super Cube Solver and compare solutions:

σ = (CKEFM) = NF SR ND NF2 ND' NF2 R' NF2 ND NF2 ND' L NF' (13 moves)

Orbit Solver – Corner-Centers 5-Cycle

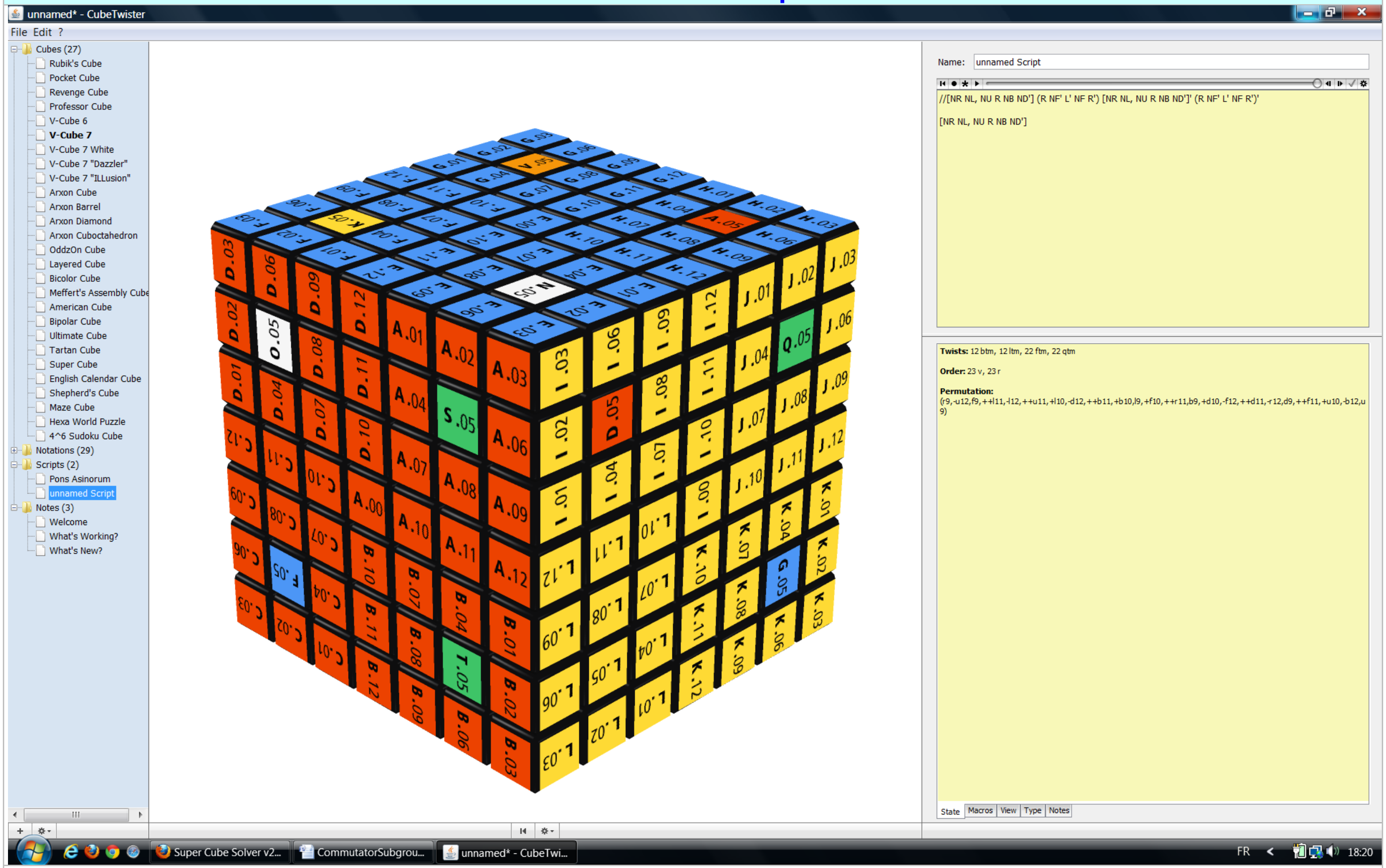
http://www.randelshofer.ch/rubik/virtualcubes/vcube7/7x_scripts/7x_super_cube_solver/index_enVE.html



σ = (CKEFM) = NF SR ND NF2 ND' NF2 R' NF2 ND NF2 ND' L NF' (13 moves)

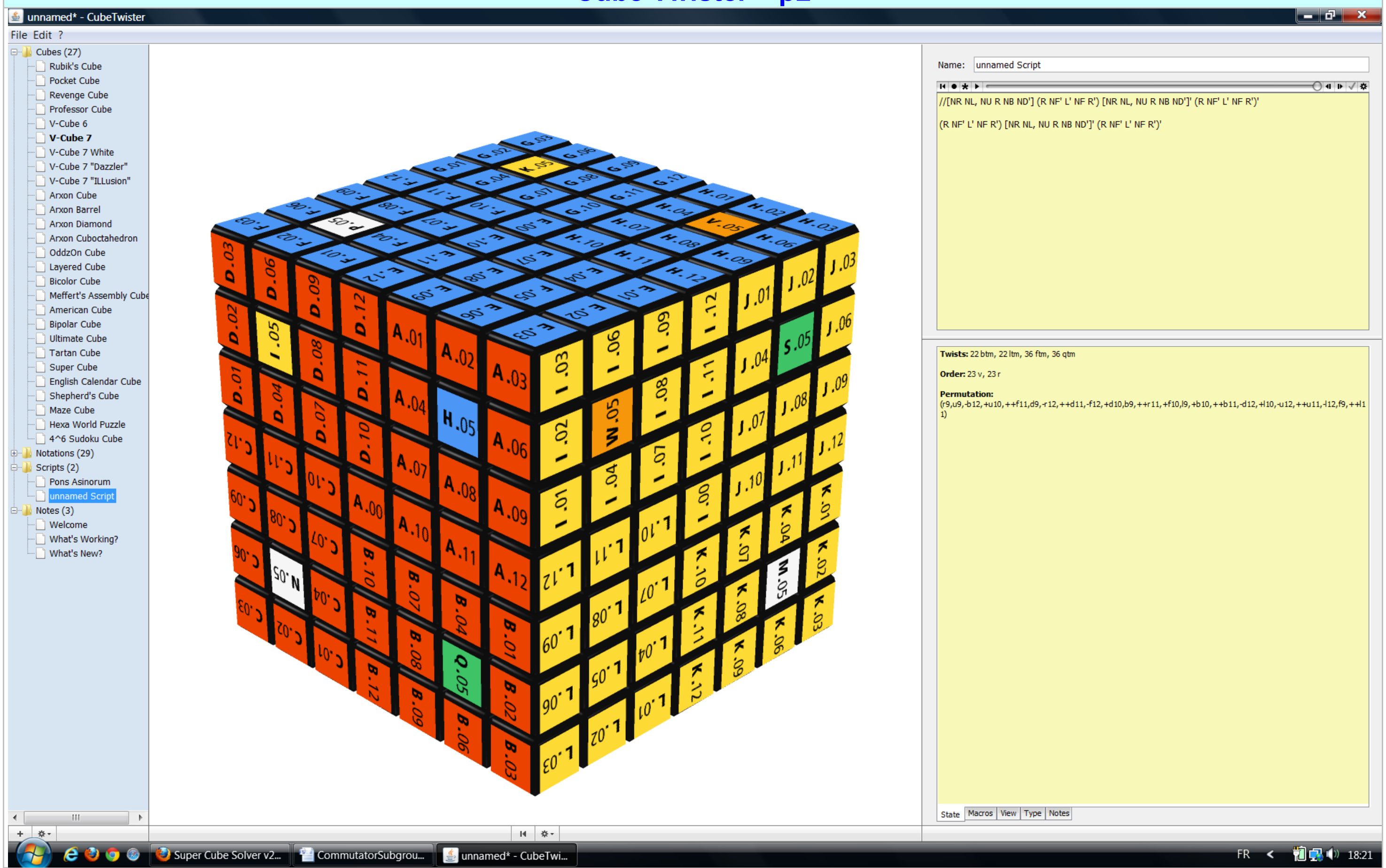
*SSE Notation

Cube Twister – ρ1



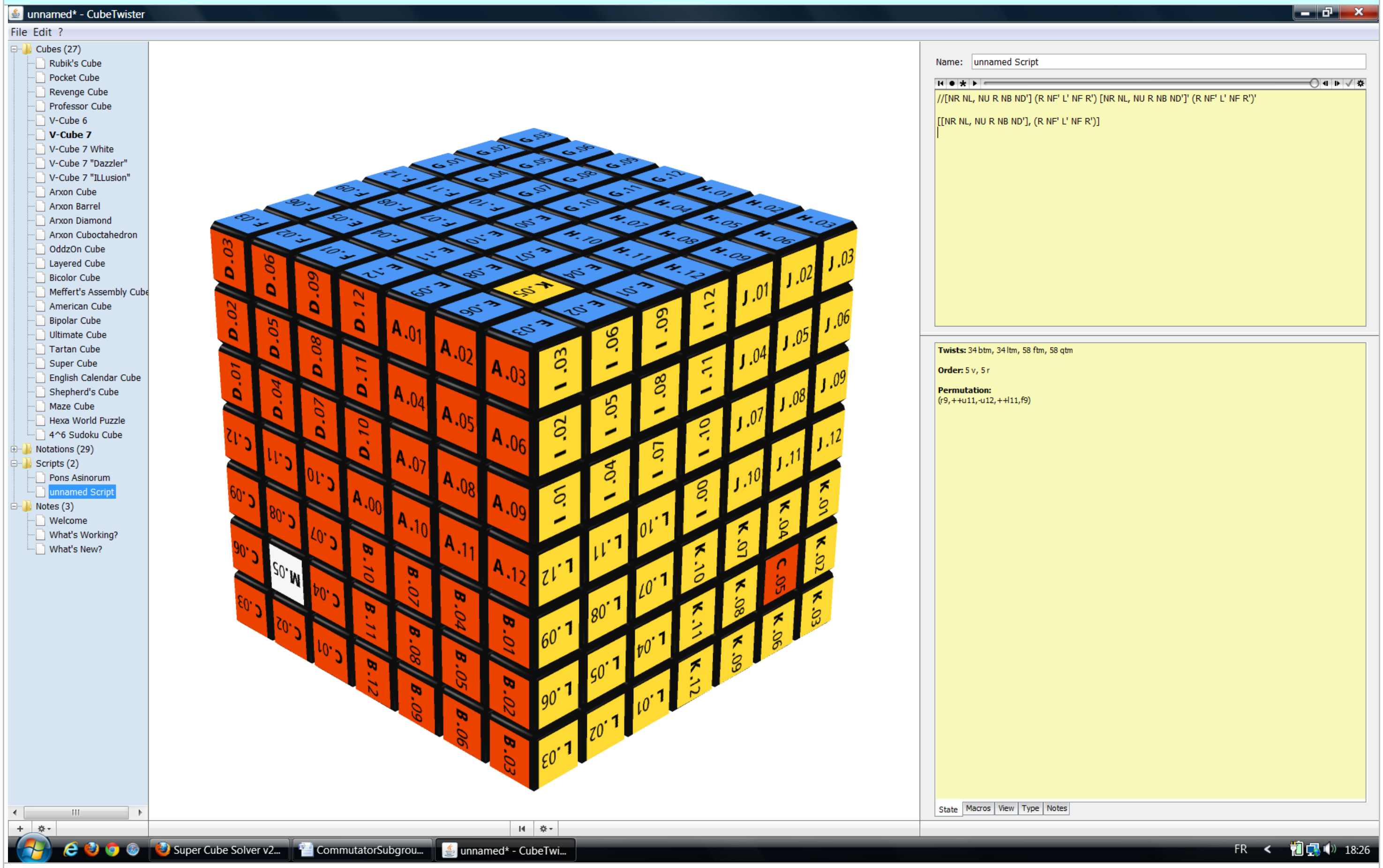
ρ1 = [NR NL, NU R NB ND']

Cube Twister – ρ2



ρ2 = τ·ρ1'·τ' = (R NF' L' NF R') [NR NL, NU R NB ND']' (R NF' L' NF R')'

Cube Twister – σ



σ = ρ1·ρ2 = [NR NL, NU R NB ND'] (R NF' L' NF R') [NR NL, NU R NB ND']' (R NF' L' NF R')'

σ = [ρ1, τ] = [[NR NL, NU R NB ND'], (R NF' L' NF R')]

Semi-Commutators

A commutator is generally defined as: [A, B] = AB·B'A', where A and B are sequences of moves, representing permutations of pieces on a cube.

It has been proved that A_n, the alternating group on *n* items, *n* >= 5, consists *entirely* of commutators, so that any permutation of A_n can be represented by a commutator or by a product of commutators, where the two representations are strictly equivalent, at least from a theoretical standpoint.

It may, however, be of interest to find products of commutators that give short sequences of moves, at least for products of a few commutators. For this to happen, there must be move cancellation between consecutive commutators. If a *maximum* number of moves could be cancelled out this way, a commutator-like expression may eventually be obtained, which may be called a *semi-commutator* – a better name is still to be found, btw.

The structure of such a semi-commutator depends on a number of *variables* and on the *direction* of enclosing brackets. Notice that this is more of a personal notation than something to be widely used…

Examples below are given for 3 and 4 variables and can be easily extended to a higher number of variables.

3 variables

Expressions of semi-commutators

[X, YZ] = X·YZ·X'·Z'Y' (commutator)
[X, YZ[= X·YZ·X'·Y'Z' (semi-commutator)
]XY, Z] = XY·Z·X'Y'·Z' (semi-commutator)
]XY, Z] = [X, YZ[

Semi-commutators as products of 2 commutators (move cancellations shown in red)

[X, YZ[= [X, YZ] [Y, Z] = X·YZ·X'·Z'Y'·Y·Z·Y'·Z' = X·YZ·X'·Z'Y'
]XY, Z] = [X, Y] [YX, Z] = X·Y·X'·Y'·YX·Z·X'Y'·Z' = XY·Z·X'Y'·Z'

Inverses of semi-commutators

[X, YZ[' =]ZY, X]
]XY, Z]' = [Z, YX]

4 variables

Expressions of semi-commutators

[XY, ZP] = XY·ZP·Y'X'·P'Z' (commutator)
[XY, ZP[= XY·ZP·Y'X'·Z'P' (semi-commutator)
]XY, ZP] = XY·ZP·X'Y'·P'Z' (semi-commutator)
]XY, ZP[= XY·ZP·X'Y'·Z'P' (semi-commutator)

Semi-commutators as products of 2 or 3 commutators (move cancellations shown in red)

[XY, ZP[= [XY, ZP] [Z, P] = XY·ZP·Y'X'·P'Z'·Z·P·Z'P' = XY·ZP·Y'X'·Z'P'
]XY, ZP] = [X, Y] [YX, ZP] = X·Y·X'·Y'·YX·ZP·X'Y'·P'Z' = XY·ZP·X'Y'·P'Z'
]XY, ZP[= [X, Y] [YX, ZP] [Z, P] = X·Y·X'·Y'·YX·ZP·X'Y'·P'Z'·Z·P·Z'·P' = XY·ZP·X'Y'·Z'P'

Inverses of semi-commutators

[XY, ZP[' =]PZ, XY]
]XY, ZP]' = [ZP, YX[
]XY, ZP[' =]PZ, YX[

Semi-commutators may be included in a brute-force search, when searching for algorithms by sweeping variables that take values in a set of basic moves, until swept permutation and goal permutation match, like in this example, where 8 variables are used:

[XYZPQ, AEG] = XYZPQ·AEG·Q'P'Z'Y'X'·G'E'A' (commutator)
[XYZPQ, AEG[= XYZPQ·AEG·Q'P'Z'Y'X'·A'E'G' (semi-commutator)
]XYZPQ, AEG] = XYZPQ·AEG·X'Y'Z'P'Q'·G'E'A' (semi-commutator)
]XYZPQ, AEG[= XYZPQ·AEG·X'Y'Z'P'Q'·A'E'G' (semi-commutator)

Semi-Commutator Example

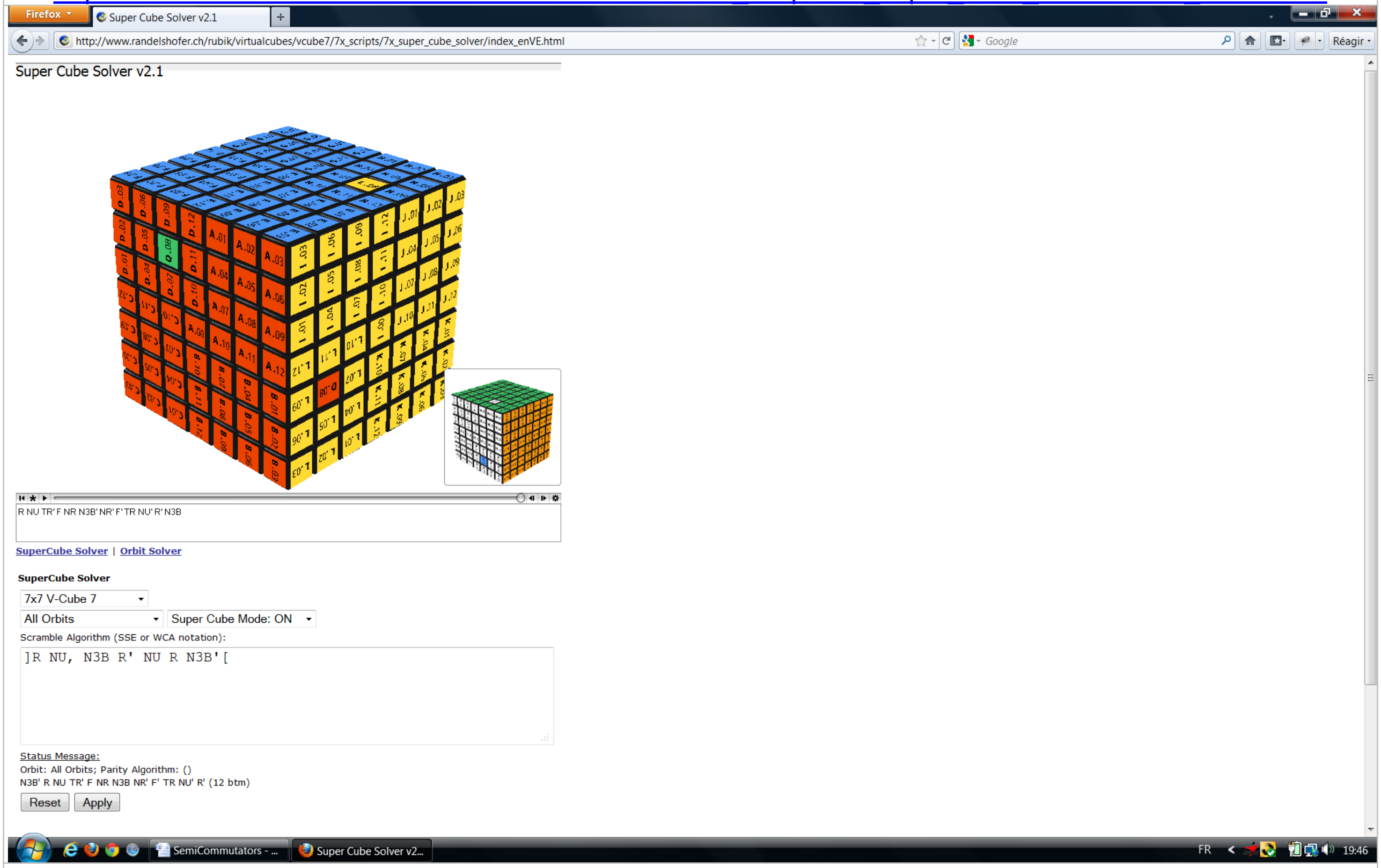
A short 5-cycle of edge-centers has been obtained from the following semi-commutator:

]R NU, N3B R' NU R N3B'[= R NU N3B R' NU R N3B' R' NU' N3B' R NU' R' N3B (14 moves)

We can search for a shorter algorithm using Super Cube Solver and compare solutions:

[N3B', R NU TR' F NR] = N3B' R NU TR' F NR N3B NR' F' TR NU' R' (12 moves)

Notice that move TR is the combination of moves R and NR, that is: TR = R NR, thus giving a shorter solution.

SuperCube Solver – Edge-Centers 5-Cycle
http://www.randelshofer.ch/rubik/virtualcubes/vcube7/7x_scripts/7x_super_cube_solver/index_enVE.html

Scramble Algorithm
]R NU, N3B R' NU R N3B'[= R NU N3B R' NU R N3B' R' NU' N3B' R NU' R' N3B
Solution Algorithm
[N3B', R NU TR' F NR] = N3B' R NU TR' F NR N3B NR' F' TR NU' R'

Symmetric Commutators

Commutators (or semi-commutators) show structures that are symmetric in nature. If we consider, for example, the following commutator [A, B] of 5 variables [X Y, Z P Q], written as:

[A, B] = [X Y, Z P Q] = X Y·Z P Q·Y' X'·Q' P' Z' = (X Y)·(Z P Q)·(X Y)'·(Z P Q)' = A·B·A'·B'

we can see that the second half of this expression is simply the composition of the inverses of A and B.

Knowing that a cube has a set of 48 symmetries, we can further expand the concept of this 'plain' commutator to the 'symmetric' commutator, where the inverses of A and B are replaced with As and Bs, being the inverses of their respectives transformations by any of the 48 cube symmetries, that is:

[A, B]s = [X Y, Z P Q]s = X Y·Z P Q·Ys' Xs'·Qs' Ps' Zs' = (X Y)·(Z P Q)·(Xs Ys)'·(Zs Ps Qs)' = A·B·As'·Bs'

In this notation, subscript 's' indicates that symmetry has been applied to the the second half of the expression.

A plain commutator is then just a particular case of a symmetric commutator, for which the applied symmetry is simply the 'Identity' symetry:

F→F
R→R
U→U
L→L
D→D
B→B

The concept of symmetric commutors can even be further expanded to symmetric *semi*-commutators as follows:

]X Y, Z P Q[s = X Y·Z P Q·Xs' Ys'·Zs' Ps' Qs'
]X Y, Z P Q]s = X Y·Z P Q·Xs' Ys'·Qs' Ps' Zs'
[X Y, Z P Q]s = X Y·Z P Q·Ys' Xs'·Qs' Ps' Zs'

It is already known that plain commutators work well in cases where only a few cube pieces are permuted. They are generally of less practical use for solving cube positions with many permuted pieces, though. But, if a scrambled cube shows a symmetric pattern, chances are good that a symmetric commutator could be found that may eventually solve it.

Symmetric Commutator Examples

Symmetric commutators may be used instead of plain commutators in difficult cases, or for finding alternate (symmetric) solution algorithms to already known ones.

As an example, we will search for an alternate algorithm to the hardest distance-20 position of a 3x3x3 cube, using symmetric commutators.

According to the 'God's Number is 20' paper, the following position was the hardest to their programs to solve:

F U' F2 D' B U R' F' L D' R' U' L U B' D2 R' F U2 D2

The algorithm itself doesn’t show any obvious symmetry, so we first have to search for symmetric cube positions, if any.

The position shows a symmetry about the F – B axis, so that a half-turn cube rotation (by move CF2) gives another position which is equivalent to the initial one.

Using Cube Explorer 5.00s, all optimal solutions to this algorithm, plus its 19 shifted versions, have been found. From the list, a 18-move algorithm was extracted that presents a symmetric commutator structure:

R' L·D2 U' F' L D U2 F'·R' L·F D2 U' R' F D U2

This algorithm can be rewritten as:

[R' L, D2 U' F' L D U2 F']s

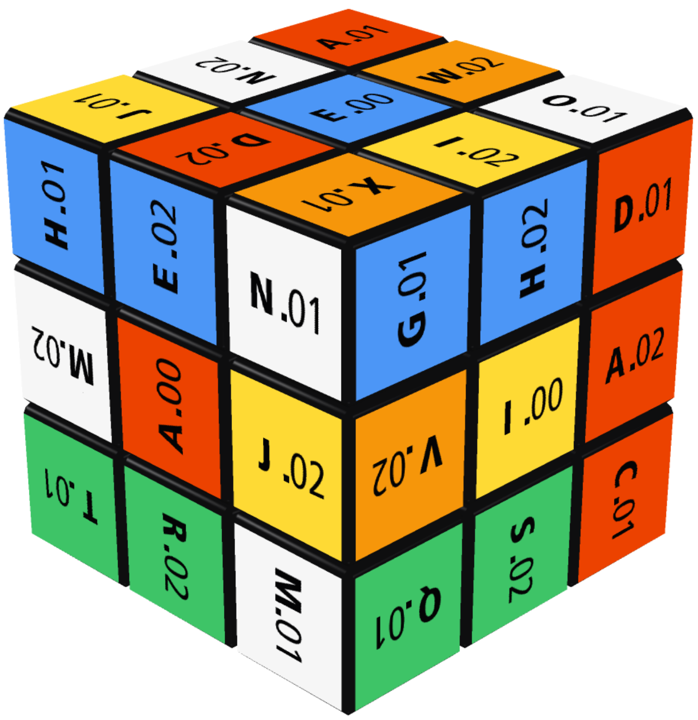
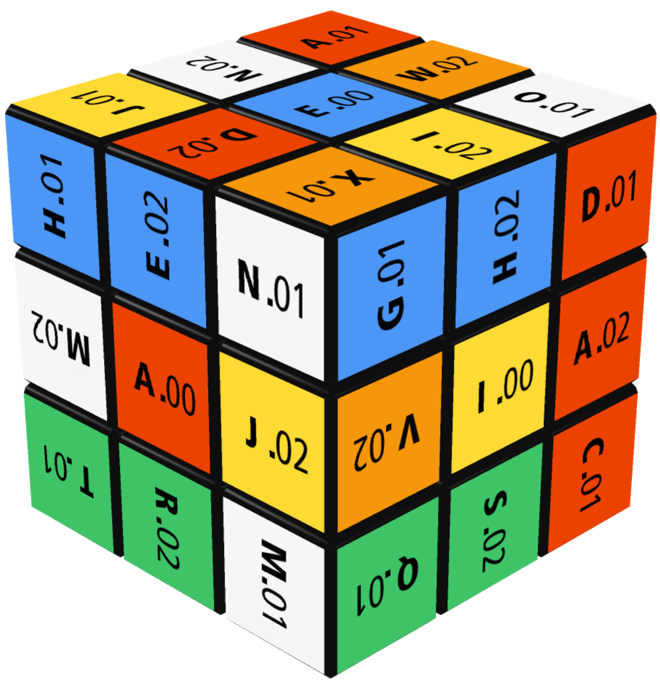
where symmetry CF2 has been applied to the second half, as follows:

F→F
R→L
U→D
L→R
D→U
B→B

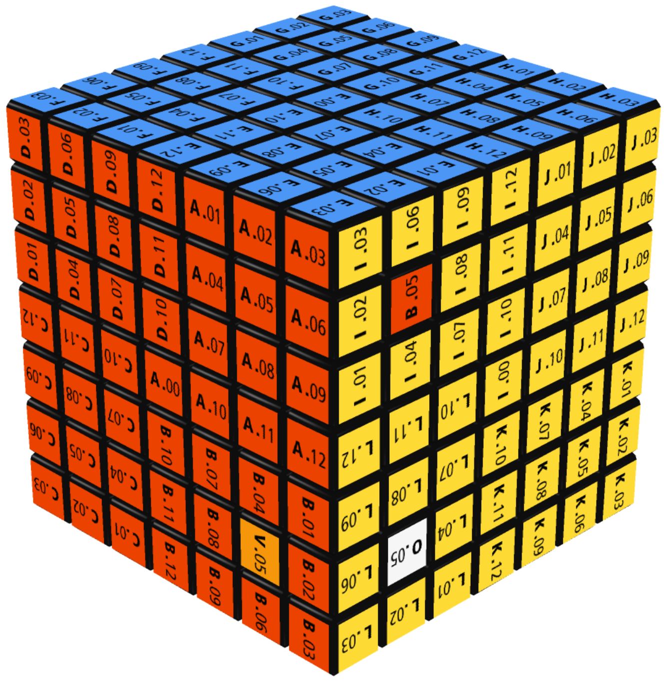
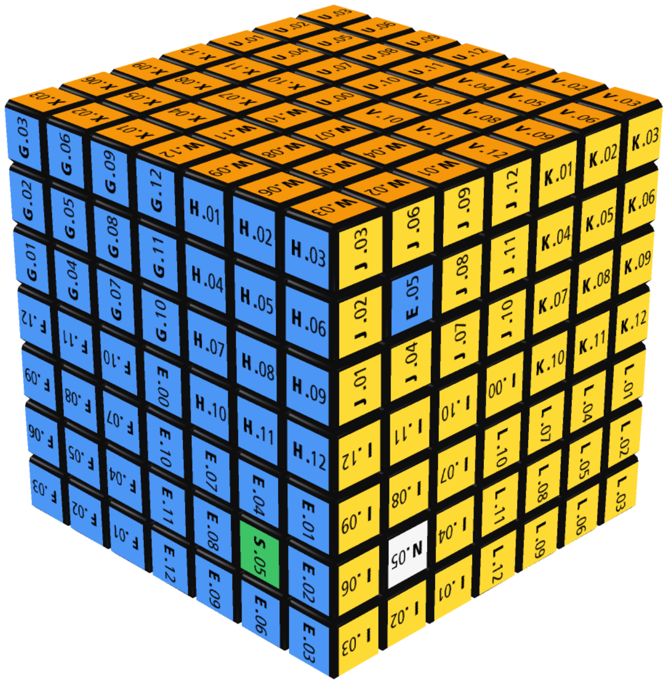
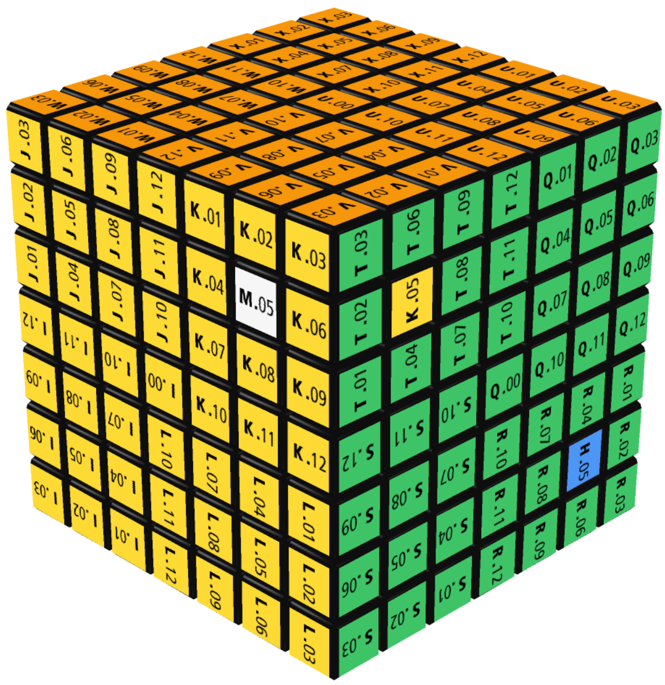
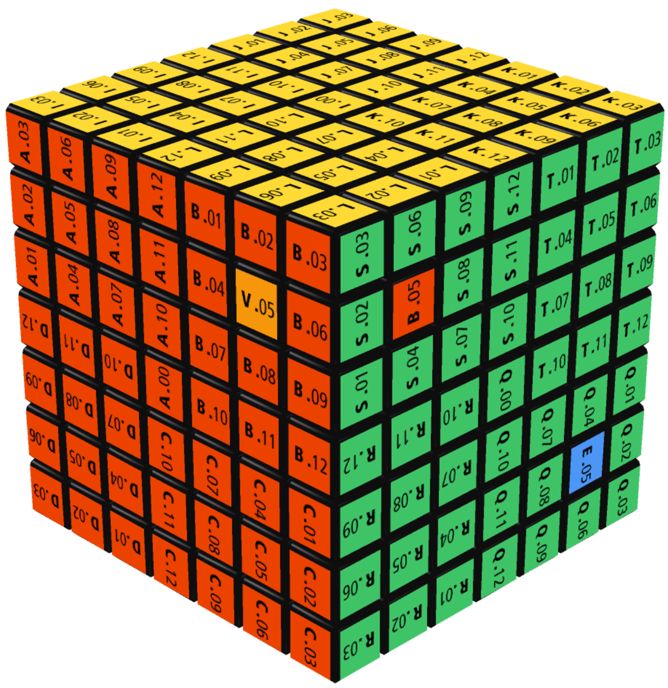
Using Algorithm Finder Lite, this 18-move algorithm has been shifted, conjugated and transformed by symmetry to give the following symmetric solutions to the initial (unsymmetric) algorithm, where SR = R L':

B D' L2 R F D' L' R2 F SU F' L2 R U F' L' R2 U B'
F L' U2 D F L' U' D2 F SR F' U2 D R F' U' D2 R F'
F D' R2 L B D' R' L2 B SU B' R2 L U B' R' L2 U F'
B R' U2 D B R' U' D2 B SR' B' U2 D L B' U' D2 L B'
F' U R2 L' B' U R L2 B' SU B R2 L' D' B R L2 D' F
B' R D2 U' B' R D U2 B' SR B D2 U' L' B D U2 L' B
B' U L2 R' F' U L R2 F' SU F L2 R' D' F L R2 D' B
F' L D2 U' F' L D U2 F' SR' F D2 U' R' F D U2 R' F

All solutions shown are 20-move algorithms in HTM and 19-move algorithms in STM.

Symmetric Commutator – Example 1	
3x3x3 Cube – Hardest Distance-20 Position	
	
Unsymmetric Algorithm	Symmetric Algorithm (s = CF2)
F U' F2 D' B U R' F' L D' R' U' L U B' D2 R' F U2 D2	F' L D2 U' F' L D U2 F' SR' F D2 U' R' F D U2 R' F

Symmetric commutators may also be used for permuting a few cube pieces, although plain commutators will generally provide shorter solutions, as shown below for the case of corner-center 5-cycle.

Symmetric Commutators	**Commutators**
[NR, NU' NR NU R[s (10 moves)	NF' D NF ND' NF' D' NF' ND NF2 (9 moves)
[NR, ND' NR' ND D2[s (10 moves)	U D NR' ND' NR U' D' NR ND NR' (10 moves)
Symmetric Commutator – Example 2	
7x7x7 Cube – Permutation (BIOLV) – Corner-Center 5-Cycle	
	
[NR, NU' NR NU R[s (s = CR')	
Original Position	Equivalent Position
NR NU' NR NU R NR' NF NR' NF' R'	CR' NR NU' NR NU R NR' NF NR' NF' R'
7x7x7 Cube – Permutation (HRMKT) – Corner-Center 5-Cycle	
	
[NR, ND' NR' ND D2[s (s = CU')	
Original Position	Equivalent Position
NR ND' NR' ND D2 NB' ND NB ND' D2	CU' NR ND' NR' ND D2 NB' ND NB ND' D2

Generalized Commutators

Any permutation of A_n can be represented by a commutator or by a product of commutators, where the two representations are strictly equivalent, at least from a theoretical standpoint.

It may, however, be of interest to find products of commutators that give short sequences of moves, at least for products of a few commutators. For this to happen, there must be move cancellation between *some* consecutive commutators. In the case where there is move cancellation between *all* consecutive commutators in the product, we get a special case which may be considered as a *generalized* commutator, written as:

[X, Y] = X Y X' Y'
[X, Y, Z] = [X, Y] [Y, Z] = X Y X' Y' Y Z Y' Z' = X Y X' Z Y' Z'
[X, Y, Z, P] = [X, Y] [Y, Z] [Z, P] = X Y X' Y' Y Z Y' Z' Z P Z' P' = X Y X' Z Y' P Z' P'
[X, Y, Z, P, Q] = [X, Y] [Y, Z] [Z, P] [P, Q] = X Y X' Y' Y Z Y' Z' Z P Z' P' P Q P' Q' = X Y X' Z Y' P Z' Q P Q'