JAPLA 2006.Feb.25　　sudoku22e.jsw

2006.2.04 ～ 2.20

数独　と　Ｊ　　（その２）

難問　ＡＰＬ　Clark の　問題　及び　易経

中野嘉弘（札幌市 83才）　　山下紀幸（横浜市 81才）　西川利男（柏市）JAPLA会長

FAX 専 011-588-3354　　T・F 045-851-3721　　T・F 047-163-0364
yoshihiro@river.ocn.ne.jp　　Toshio.Nishikawa@kiu.ne.jp
nakano@mta.biglobe.ne.jp

Sudoku の解法、 前報の　Hui を超えた　Super Hui の話に引き続き、 ＡＰＬでの Adrian Smith 紹介の難問（ 原著 John Clark や Ellis Morgan ）を攻略した話をする。 難問の根源は莫大なる複数解の存在である。　　Clark の問題では、それは、なんと １万５千 ～　１６万　！！！ を超えた。　本稿は　Second Hui の話が主流で、終わり「目出たし」となりそうだったが、最後に「易経」問題が到来。　即応の解は得た。

は　し　が　き

ロートルもナウイ事をやる。　　欧米にも人気のパズル Sudoku 「数独」に関する「数独とＪ（その１）」（文献１）の続報である。　前報は、Vector 誌の Mr. Hui の論文（文献２）を入手する以前に、肝心の事は殆ど終わっていた。
また、Hui は サイズ ９ｘ９で、正方形 BOX に限定し、且つ、解く話のみ！　であるから、前報は、 Hui を材料にしても、実は Hui を超える　Super Hui の話になった。
その目次を再録して、 本稿との繋がりを示そう。

目次　（その１）

不定形 BOX には、クリスマスツリーなどの他、　例えば以下のもの
Star Sudoku: Thu 22-Dec-2005　　medium
Scary Angel Sudoku:Wed 21-Dec-2005　　hard
Church window Sudoku:Tue 20-Dec-2005　　hard
Angel:Fri 30-Dec-2005　　hard
To replace the duplicate:Thu 29-Dec-2005　　very hard
Christmas tree Sudoku:Fri 23-Dec-2005　　very hard
Take down the tree:Sat 31-Dec-2005　　very hard
まるで、教会のステンドグラスもどきの多様性である。
以上の話題は、（その１）で、すでに処理すみのである。

目次　　（その２）

５．　複数解の話題　　６．　候補選び　中野の　Hybrid 法
７．　易経 と 山下の 即応解　　８．　易経　の　プログラムリスト

今までに、西川は、 Hui のＪプログラムのアルゴリズムの分析を行った（文献３）。これは重要な話題である。

また、 Hui の 論文掲載の VECTOR 誌内で、並んで、 Adrian Smith が ＡＰＬでの、パソコン解法の例を紹介したものがあった（文献４）。 Hui の論文の３倍くらいの分量がある。 このＡＰＬ流の解法は、神秘的魅力も感じさせたが、大変な難問で、前報では、紙面の関係もあり、浅くしか触れられなかった。 今回はそれを扱う。

更に、高次のものでは、Ellis Morgan の ２７次 を超えた、我らの３６次の例をも紹介する。

報告記事の順序は、研究の時系列ではなくて、説明・理解のし易いであろう順序にしてある。 後に書いてあることが、実は先に済んでいた事もある。

なお、長大記事は末尾に纏めることがある（特に電子版では）、御了承下さい。

参考の為に、用語やルール等を略述して置く。 基本形 (classic) の場合

０）数字 1 ～ 9 を空所に入れるパズルである。

１） 9 x 9 ケの grid （格子）の cell （マス目）と、その中に サイズ 3 x 3 の subgrids （box とか region とか block と呼ぶ）がある。

既に数字が入っている処がある（givens とか clues = 手がかり、糸口 と呼ぶ）。

２）各行 row、各列 column、各 box の空所（ empty cells、 missing numbers、open とか）に、重複しないように（only once, singly）数字を入れよ（fill）。

と云う簡単なルールで、 「数独 single numbers」 の名前の起源である。

この数字とは、単に「目印、マーク」（例えばトランプの）であって、数学演算の対象では無い。 アルファベット や イロハニホヘト等でも同じことだ。

その点、魔方陣などより簡単である。 （発展形については、後述する。）

（その２）の原稿を殆ど、書き終わった頃、「志村 e-mail : 易経」として、超 Hui の解法の海外での試みを知らされた。 <godspiral2000@yahoo.ca> （文献９）

１． サイズ ３６ｘ３６ の数独問題

「数独、ナンプレ」の問題集は Hui が扱った classic な 9x9 次正方形が殆どであるから、中野は ２５次 が或いは世界最高次かと考え、問題の作成・解答の例を前法６．節 と １２．節に与えた。

しかし、直前の「はしがき」の末尾で触れた Adrian Smith の紹介中の Ellis Morgan の プログラム Code 中に 27x27 の記事があり、squares bordered とあるが、BOX を 9x9 とするならば合点が行かぬ。 9x9 BOX 内の cell は ８１ケある。

しかるに row または column内の cell は２７ケ であるから、マッチしない。

マッチする為には、3 x 9 の矩形 BOX を用意せねばならぬが、その気配は見えぬ。

また、27次の問題の実例も見えぬ。 よって、 27次 は幻で 、中野の 25次が、やはり最高次かと思われるが、念の為、さらに ３６次 の作成問題 と 解答例を示しておく。

（長大のため、最寄りの見開きページにて示す）

２． 西川の Hui 法 の 分析 （候補選び）

（その１）には平行して、Hui プログラムの分析記事を西川が提供した（文献３）。

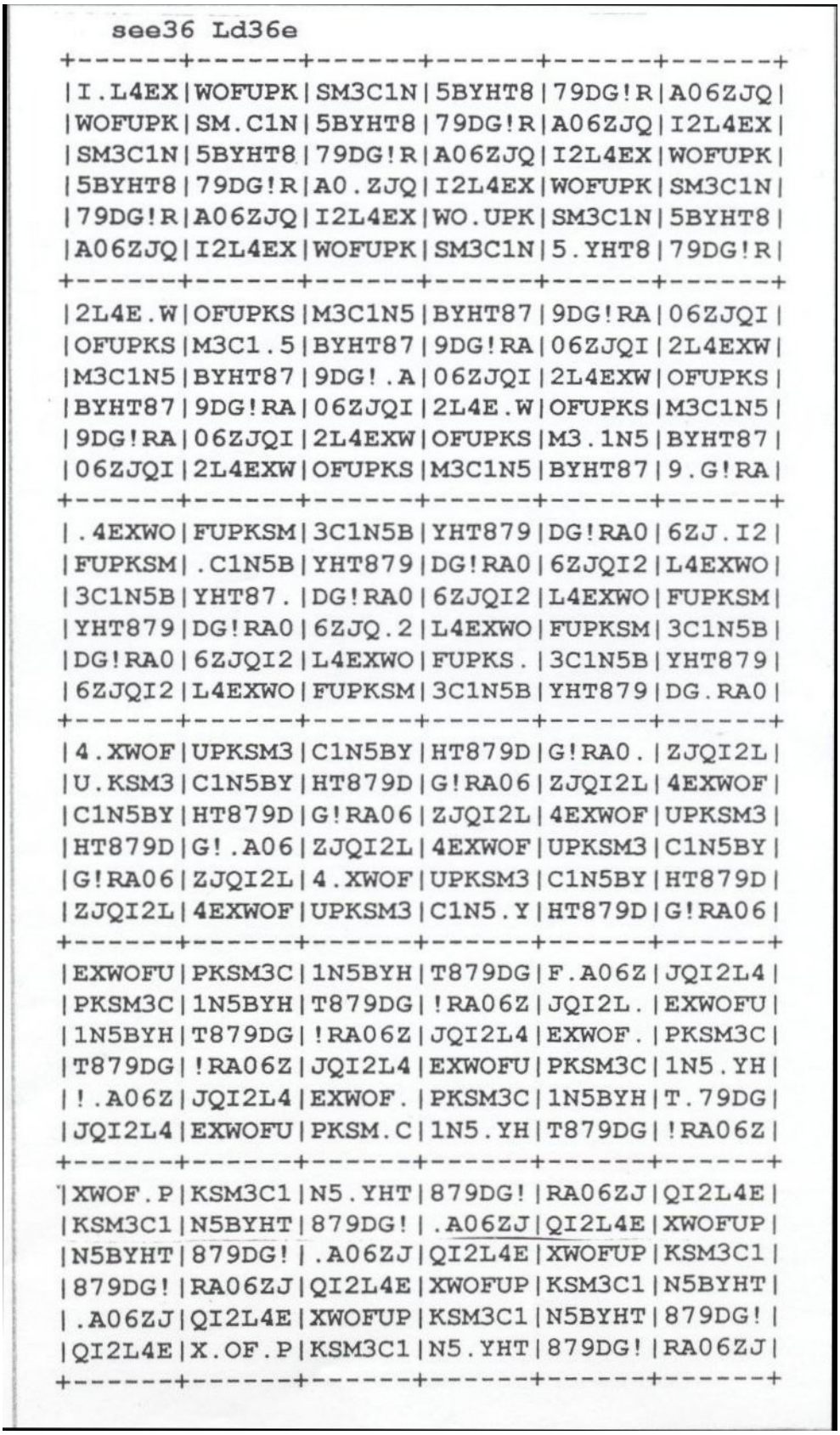
Labs システムによるとあるが、二老人は不幸にして、肝心の Labs システムに無知なので直接の御利益は得られなかったが、 Candidate （候補者）関数の概念は、中野が、Hui 論文を知る以前にやっていた数独の Hybrid 解法と同好なのに感激した。

Hybrid とは、解法の全部をＰＣに任せるのでは無くて、人間が鉛筆をもってトライする如く、いわば手動で cell へ候補を入力しながら、進行させる事をも可とする hybrid

混成解法 である。　grid や BOX や cell の構成が、Hui の如く classic な スタイル から解放されるので、大変 フレキシブルである。　クリスマスツリー的な不定形に向く。
その代わり、「２次方程式の根の公式に数値を代入して、ハイ終わり」的には行かない。
（以下、話は６．節に続く。　次ページは、１節. の　サイズ　３６ｘ３６　の図）

(1. 節の続き)　　最大サイズ　３６ｘ３６　の　数独　問題　（中野）

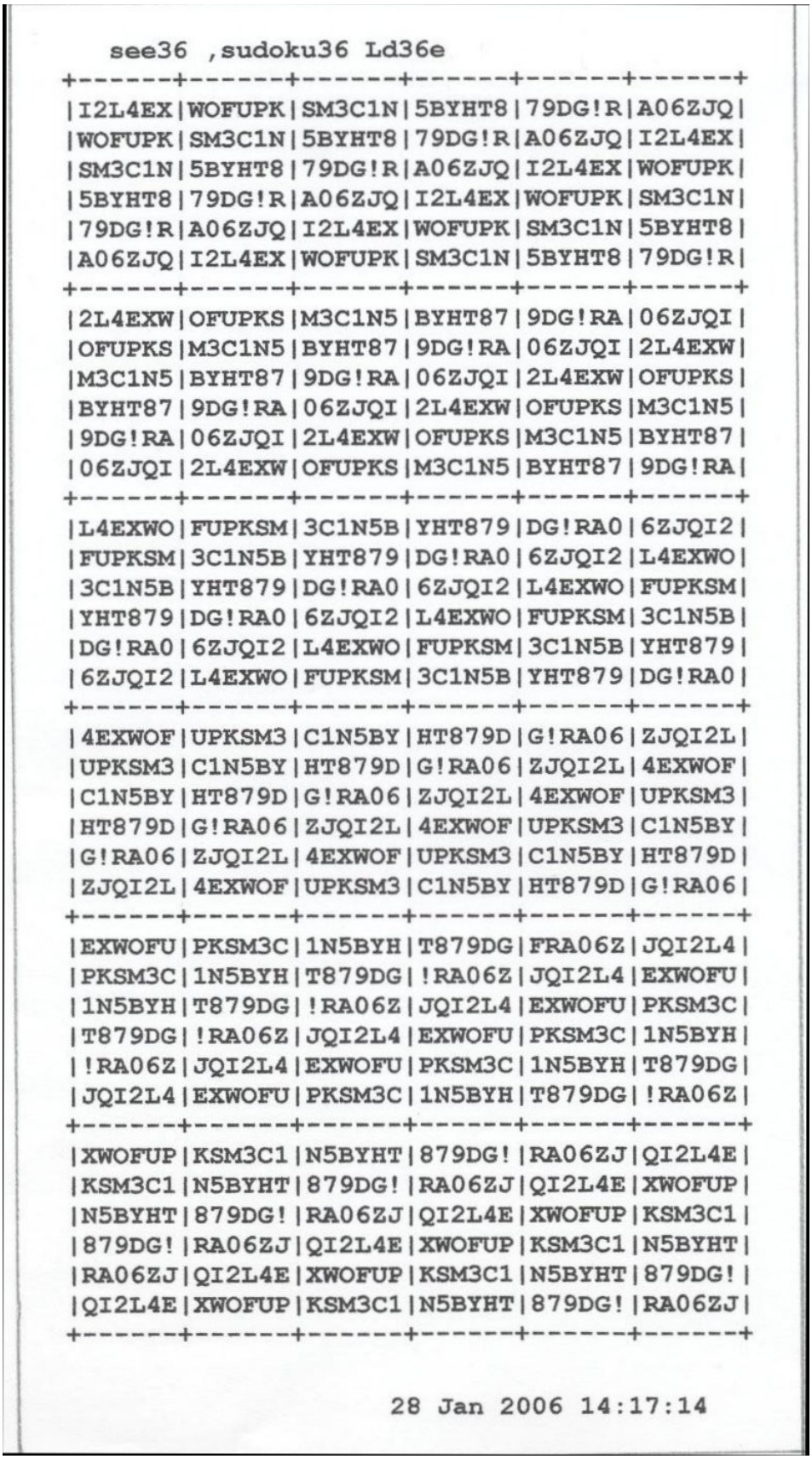
関数　**see36**　は　このサイズに専用



```
   see36 Ld36e
+------+------+------+------+------+------+
|I.L4EX|WOFUPK|SM3C1N|5BYHT8|79DG!R|A06ZJQ|
|WOFUPK|SM.C1N|5BYHT8|79DG!R|A06ZJQ|I2L4EX|
|SM3C1N|5BYHT8|79DG!R|A06ZJQ|I2L4EX|WOFUPK|
|5BYHT8|79DG!R|A0.ZJQ|I2L4EX|WOFUPK|SM3C1N|
|79DG!R|A06ZJQ|I2L4EX|WO.UPK|SM3C1N|5BYHT8|
|A06ZJQ|I2L4EX|WOFUPK|SM3C1N|5.YHT8|79DG!R|
+------+------+------+------+------+------+
|2L4E.W|OFUPKS|M3C1N5|BYHT87|9DG!RA|06ZJQI|
|OFUPKS|M3C1.5|BYHT87|9DG!RA|06ZJQI|2L4EXW|
|M3C1N5|BYHT87|9DG!.A|06ZJQI|2L4EXW|OFUPKS|
|BYHT87|9DG!RA|06ZJQI|2L4E.W|OFUPKS|M3C1N5|
|9DG!RA|06ZJQI|2L4EXW|OFUPKS|M3.1N5|BYHT87|
|06ZJQI|2L4EXW|OFUPKS|M3C1N5|BYHT87|9.G!RA|
+------+------+------+------+------+------+
|.4EXWO|FUPKSM|3C1N5B|YHT879|DG!RA0|6ZJ.I2|
|FUPKSM|.C1N5B|YHT879|DG!RA0|6ZJQI2|L4EXWO|
|3C1N5B|YHT87.|DG!RA0|6ZJQI2|L4EXWO|FUPKSM|
|YHT879|DG!RA0|6ZJQ.2|L4EXWO|FUPKSM|3C1N5B|
|DG!RA0|6ZJQI2|L4EXWO|FUPKS.|3C1N5B|YHT879|
|6ZJQI2|L4EXWO|FUPKSM|3C1N5B|YHT879|DG.RA0|
+------+------+------+------+------+------+
|4.XWOF|UPKSM3|C1N5BY|HT879D|G!RA0.|ZJQI2L|
|U.KSM3|C1N5BY|HT879D|G!RA06|ZJQI2L|4EXWOF|
|C1N5BY|HT879D|G!RA06|ZJQI2L|4EXWOF|UPKSM3|
|HT879D|G!.A06|ZJQI2L|4EXWOF|UPKSM3|C1N5BY|
|G!RA06|ZJQI2L|4.XWOF|UPKSM3|C1N5BY|HT879D|
|ZJQI2L|4EXWOF|UPKSM3|C1N5.Y|HT879D|G!RA06|
+------+------+------+------+------+------+
|EXWOFU|PKSM3C|1N5BYH|T879DG|F.A06Z|JQI2L4|
|PKSM3C|1N5BYH|T879DG|!RA06Z|JQI2L.|EXWOFU|
|1N5BYH|T879DG|!RA06Z|JQI2L4|EXWOF.|PKSM3C|
|T879DG|!RA06Z|JQI2L4|EXWOFU|PKSM3C|1N5.YH|
|!.A06Z|JQI2L4|EXWOF.|PKSM3C|1N5BYH|T.79DG|
|JQI2L4|EXWOFU|PKSM.C|1N5.YH|T879DG|!RA06Z|
+------+------+------+------+------+------+
|XWOF.P|KSM3C1|N5.YHT|879DG!|RA06ZJ|QI2L4E|
|KSM3C1|N5BYHT|879DG!|.A06ZJ|QI2L4E|XWOFUP|
|N5BYHT|879DG!|.A06ZJ|QI2L4E|XWOFUP|KSM3C1|
|879DG!|RA06ZJ|QI2L4E|XWOFUP|KSM3C1|N5BYHT|
|.A06ZJ|QI2L4E|XWOFUP|KSM3C1|N5BYHT|879DG!|
|QI2L4E|X.OF.P|KSM3C1|N5.YHT|879DG!|RA06ZJ|
+------+------+------+------+------+------+
```

（1．節の続き）　　最大サイズ　３６ｘ３６　数独　の解　（中野）

関数　　**sudoku36**　も　このサイズ専用



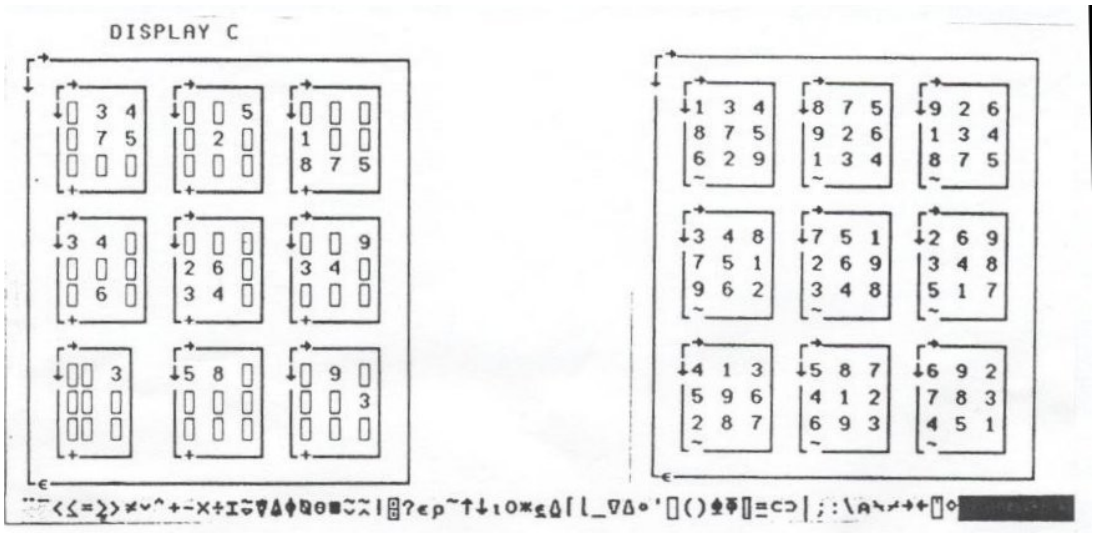
```
   see36 ,sudoku36 Ld36e
+------+------+------+------+------+------+
|I2L4EX|WOFUPK|SM3C1N|5BYHT8|79DG!R|A06ZJQ|
|WOFUPK|SM3C1N|5BYHT8|79DG!R|A06ZJQ|I2L4EX|
|SM3C1N|5BYHT8|79DG!R|A06ZJQ|I2L4EX|WOFUPK|
|5BYHT8|79DG!R|A06ZJQ|I2L4EX|WOFUPK|SM3C1N|
|79DG!R|A06ZJQ|I2L4EX|WOFUPK|SM3C1N|5BYHT8|
|A06ZJQ|I2L4EX|WOFUPK|SM3C1N|5BYHT8|79DG!R|
+------+------+------+------+------+------+
|2L4EXW|OFUPKS|M3C1N5|BYHT87|9DG!RA|06ZJQI|
|OFUPKS|M3C1N5|BYHT87|9DG!RA|06ZJQI|2L4EXW|
|M3C1N5|BYHT87|9DG!RA|06ZJQI|2L4EXW|OFUPKS|
|BYHT87|9DG!RA|06ZJQI|2L4EXW|OFUPKS|M3C1N5|
|9DG!RA|06ZJQI|2L4EXW|OFUPKS|M3C1N5|BYHT87|
|06ZJQI|2L4EXW|OFUPKS|M3C1N5|BYHT87|9DG!RA|
+------+------+------+------+------+------+
|L4EXWO|FUPKSM|3C1N5B|YHT879|DG!RA0|6ZJQI2|
|FUPKSM|3C1N5B|YHT879|DG!RA0|6ZJQI2|L4EXWO|
|3C1N5B|YHT879|DG!RA0|6ZJQI2|L4EXWO|FUPKSM|
|YHT879|DG!RA0|6ZJQI2|L4EXWO|FUPKSM|3C1N5B|
|DG!RA0|6ZJQI2|L4EXWO|FUPKSM|3C1N5B|YHT879|
|6ZJQI2|L4EXWO|FUPKSM|3C1N5B|YHT879|DG!RA0|
+------+------+------+------+------+------+
|4EXWOF|UPKSM3|C1N5BY|HT879D|G!RA06|ZJQI2L|
|UPKSM3|C1N5BY|HT879D|G!RA06|ZJQI2L|4EXWOF|
|C1N5BY|HT879D|G!RA06|ZJQI2L|4EXWOF|UPKSM3|
|HT879D|G!RA06|ZJQI2L|4EXWOF|UPKSM3|C1N5BY|
|G!RA06|ZJQI2L|4EXWOF|UPKSM3|C1N5BY|HT879D|
|ZJQI2L|4EXWOF|UPKSM3|C1N5BY|HT879D|G!RA06|
+------+------+------+------+------+------+
|EXWOFU|PKSM3C|1N5BYH|T879DG|FRA06Z|JQI2L4|
|PKSM3C|1N5BYH|T879DG|!RA06Z|JQI2L4|EXWOFU|
|1N5BYH|T879DG|!RA06Z|JQI2L4|EXWOFU|PKSM3C|
|T879DG|!RA06Z|JQI2L4|EXWOFU|PKSM3C|1N5BYH|
|!RA06Z|JQI2L4|EXWOFU|PKSM3C|1N5BYH|T879DG|
|JQI2L4|EXWOFU|PKSM3C|1N5BYH|T879DG|!RA06Z|
+------+------+------+------+------+------+
|XWOFUP|KSM3C1|N5BYHT|879DG!|RA06ZJ|QI2L4E|
|KSM3C1|N5BYHT|879DG!|RA06ZJ|QI2L4E|XWOFUP|
|N5BYHT|879DG!|RA06ZJ|QI2L4E|XWOFUP|KSM3C1|
|879DG!|RA06ZJ|QI2L4E|XWOFUP|KSM3C1|N5BYHT|
|RA06ZJ|QI2L4E|XWOFUP|KSM3C1|N5BYHT|879DG!|
|QI2L4E|XWOFUP|KSM3C1|N5BYHT|879DG!|RA06ZJ|
+------+------+------+------+------+------+

                    28 Jan 2006 14:17:14
```

３．　ＡＰＬの　Ｃｌａｒｋの難問題

Vector誌に Hui と並んで　Adrian Smith の　”　Sudoku with Dyalog APL from

John Clark & Ellis Morgan " なる紹介記事がある（文献４）。　Hui の論文の３倍も長く、神秘的である。　問題の例は、Times of London　から採ったものとか。
難度は　EASY, MEDIUM, HARD and VERY_HARD の最高ランクのものだ（結果的に）。
　サイズ 9x9 、BOX は 3x3 正方形の古典型であるから、 Hui 流で一発求解と思いきや、返り討ち！　　81 cells　のうち、given（指定済み）が 25 ケ処、 open（空所）が 56ケ処は、 medium　と　hard の中間の難易度に過ぎない！
しかし、前報１０．節で中野は、問題を勝手に直し、given を増し、 open を減らし同数程度に変更して（問題 apl9941）、辛うじて解らしきものを得た（前報 p.15）。
西川は ＡＰＬ　code　中の　BACKTRACKING 法と云う神秘的な名前に着目して、早速ＡＰＬ２を用いてトライ開始したが、そのＡＰＬシステムが不適か？　結果は残念。
中野は、本来の Dyalog APL を英国に発注したが、未だに、「なしのつぶて」。
やむなく、問題の SOURCE と　SOLUTION （既知）を「三枝ＡＰＬ２（文献５）」で印字（のみ！）して、憂さ晴らししていた。　図は下掲。
　そうこうしている中に、山下が、解けたと報告して来た（文献７）。
それは、そのまま、次節４．に示す。



４．　山下の　Ｃｌａｒｋ　問題解

（見開き　４ページ分　を　次頁以下に示す。）

山下の clark 問題解法

Ms. Hui の提案した有効フレーズ中の (p5)g を下記の clark 問題に適用すると，下記のような2案が示される．

```
see clark,g clark
```

.34	..5	...
.75	.2.	1..
...	...	875
34.	...	..9
...	26.	34.
.6.	34.	...
..3	58.	.9.
...	...	..3
...	...	...

.34	.15	9..
.75	.2.	134
...	...	875
34.	.5.	..9
...	26.	34.
.6.	34.	...
..3	58.	.9.
...	...	..3
...	...	...

.34	.75	9..
.75	.2.	134
...	...	875
34.	.5.	..9
...	26.	34.
.6.	34.	...
..3	58.	.9.
...	...	..3
...	...	...

そこで，2案のうち，右の方が正解に辿り着く可能性が高いと想定し，右の方を選択する関数 sdky を作る．

```
g=: guess @:(ok # ]) @:assign
sdky=: 3 :'{: g y.'
```

sdky 関数を繰り返し作用させて次々に空所を埋めて行くと，最後には単一解に到達することが判った．

```
see clark,(z=.sdky ^:(13+i.3) clark)
```

.34	..5	...
.75	.2.	1..
...	...	875
34.	...	..9
...	26.	34.
.6.	34.	...
..3	58.	.9.
...	...	..3
...	...	...

834	175	962
975	826	134
621	934	875
348	751	629
597	268	341
162	349	758
413	587	.96
7..	..2	..3
...	..3	..7

834	175	962
975	826	134
621	934	875
348	751	629
597	268	341
162	349	758
413	587	296
78.	..2	..3
2..	..3	..7

834	175	962
975	826	134
621	934	875
348	751	629
597	268	341
162	349	758
413	587	296
786	492	513
259	613	487

最終単一解

なお，上表の空所のある途中段階の出力に sudoku 関数を適用させると，次頁の表のように複数解が求められる．

```
see clark,(z=.sudoku sdky ^:(12) clark)
```

.34	..5	...	834	175	962	834	175	962	834	175	962
.75	.2.	1..	975	826	134	975	826	134	975	826	134
...	...	875	621	934	875	621	934	875	621	934	875
34.	...	..9	348	751	629	348	751	629	348	751	629
...	26.	34.	597	268	341	597	268	341	597	268	341
.6.	34.	...	162	349	758	162	349	758	162	349	758
..3	58.	.9.	213	587	496	413	587	296	413	587	296
...	...	..3	786	492	513	759	612	483	786	492	513
...	...	...	459	613	287	286	493	517	259	613	487

鏡玻解　　最終単一解

次に，clark 問題の空所を次々に増やして行った場合でも解が求まるかどうかを考察する．

条件は次の5種類とする

(1) clarkV --- 第1行目を全部0とする

(2) clark V1 --- 第1，2行目を全部0とする

(3) clark V2 --- 第1，2，3，7，8，9行目を全部0とする

(4) clarkV3 --- 第1，2，3，4，6，7，8，9行目を全部0とする

(5) clark V4 --- セルの6を除き，全部0とする

(6) clarkV5 --- 全ポジションを0とする．

(1).

```
see clarkv,(clv=.sdky ^:(18+i.3)clarkv)
```

...	...	...	681	475	932	681	475	932	681	475	932
.75	.2.	1..	975	823	164	975	823	164	975	823	164
...	...	875	234	196	875	234	196	875	234	196	875
34.	...	..9	342	758	619	342	758	619	342	758	619
...	26.	34.	158	269	347	158	269	347	158	269	347
.6.	34.	...	769	341	528	769	341	528	769	341	528
..3	58.	.9.	413	587	296	413	587	296	413	587	296
...	...	..3	897	612	453	897	612	453	897	612	453
...	...	...	526	934	781	526	934	781	526	934	781

Y2　　最終単一解

(2) clarkv1

```
see clarkv1,(clv1=.sdky ^:(24+i.3) clarkv1)
```

...	...	...	285	713	964	285	713	964	285	713	964
...	...	...	796	458	231	796	458	231	796	458	231
...	...	875	43.	926	875	431	926	875	431	926	875
34.	...	..9	342	875	619	342	875	619	342	875	619
...	26.	34.	517	269	348	517	269	348	517	269	348
.6.	34.	...	968	341	752	968	341	752	968	341	752
..3	58.	.9.	.73	582	.96	173	582	496	173	582	496
...	...	..3	...	.97	..3	8.4	.97	..3	854	697	123
...	...	...	..9	.34	..7	..9	.34	..7	629	134	587

最終單一解

(3) clarkv2

```
see clarkv2,(clv2=.sdky ^:(33+i.3) clarkv2)
```

...	...	...	897	653	421	897	653	421	897	653	421
...	...	...	536	412	897	536	412	897	536	412	897
...	...	...	124	987	563	124	987	563	124	987	563
34.	...	..9	342	875	619	342	875	619	342	875	619
...	26.	34.	751	269	348	751	269	348	751	269	348
.6.	34.	...	968	341	752	968	341	752	968	341	752
...	...	...	679	528	134	679	528	134	679	528	134
...	...	...	4..	...	...	483	.96	2.5	483	196	275
...	...	...	...	...	...	2.5	.34	9.6	215	734	986

最終單一解

(4) clarkv3

```
see clarkv3,(clv3=.sdky ^:(41+i.3) clarkv3)
```

...	...	...	849	527	136	849	527	136	849	527	136
...	...	...	236	198	457	236	198	457	236	198	457
...	...	...	715	643	829	715	643	829	715	643	829
...	...	...	654	731	982	654	731	982	654	731	982
...	26.	34.	987	265	341	987	265	341	987	265	341
...	...	...	321	984	765	321	984	765	321	984	765
...	...	...	598	476	21.	598	476	213	598	476	213
...	...	...	...	...	...	4..	...	...	462	319	578
...	...	...	...	...	...	...	...	...	173	852	694

最終單一解

Y3 -8-

(5) clarkV4

```
see clarkv4,(clv4=.sdky ^:(44+i.3) clarkv4)
```

...	...	...	874	695	321	874	695	321	874	695	321
...	...	...	521	483	976	521	483	976	521	483	976
...	...	...	963	271	854	963	271	854	963	271	854
...	...	...	248	759	163	248	759	163	248	759	163
...	.6.	...	719	368	542	719	368	542	719	368	542
...	...	...	635	142	79.	635	142	798	635	142	798
...	...	...	4..	937	.1.	486	937	.15	486	937	215
...	...	...	1.7	826	43.	157	826	439	157	826	439
...	...	...	3..	514	..7	39.	514	.87	392	514	687

最終単一解

(6) clarkV5

```
see clarkv5,(clv5=.sdky ^:(46+i.3) clarkv5)
```

...	...	...	987	654	321	987	654	321	987	654	321
...	...	...	654	321	987	654	321	987	654	321	987
...	...	...	321	987	654	321	987	654	321	987	654
...	...	...	879	436	215	879	436	215	879	436	215
...	...	...	235	198	746	235	198	746	235	198	746
...	...	...	416	572	893	416	572	893	416	572	893
...	...	...	793	845	162	793	845	162	793	845	162
...	...	...	5..	...	...	568	.1.	4..	568	213	479
...	...	...	...	...	...	14.	.6.	5..	142	769	538

最終単一解

なお、上記の空所のある途中段階の出力に sudoku 関数を適用させると、下記のように複数解が求められる。

```
see clarkv5,(z=.sudoku sdky ^:(46) clarkv5)
```

...	...	...	987	654	321	987	654	321	987	654	321
...	...	...	654	321	987	654	321	987	654	321	987
...	...	...	321	987	654	321	987	654	321	987	654
...	...	...	879	436	215	879	436	215	879	436	215
...	...	...	235	198	746	235	198	746	235	198	746
...	...	...	416	572	893	416	572	893	416	572	893
...	...	...	793	845	162	793	845	162	793	845	162
...	...	...	562	719	438	568	213	479	568	213	479
...	...	...	148	263	579	142	769	538	142	769	538

Y4 -9- 複数解 最終単一解

５．　複　　数　　解　　の 話題

５－ａ　　既　知　の　こ　と

中野は当初から、 Sudoku の問題作成の方を狙っていたが、実は、作成と解答とは表裏一体のものである。　複数解を持つ如き（下手な？）問題を作って、テスト担当の老友・山下に「解が無い」等とこぼされた。（文献１、 p.3）
これは極端としても、前節の　Clark の問題の本質も、実はこれと同じだ。
解が単一で無く、複数解を持つ事が稀では無いのだ。
この様な複数解の起きないように、必要な given （置数）を与える事が、 SuDoku 出題者の苦労する処である。　出題者は、解いて見てから、出題している筈であるが、複数解は旨く避けられるのかな？　 Jsoftwareの達人、 Roger Hui の今回の解法（文献２、２－ａ）で、この複数解を処理する事が（複数解の個数分の出力も）出来た。

前報（文献１ p.5）　４ｘ４　Ｇｒｉｄ　の　場合　に、全部が無指定 open でも、２８８通りの解があり、また、１ケのみ指定 given でも、それぞれ　７２通りの解がある事を、既に示してある。
中野の問題作成の経験からすれば、ＡＰＬの　Ｃｌａｒｋの問題も、中野同様、作成に群論を利用しているように見える。　つまり、素人の学者先生の作品だろう。　プロのメーカーなら、もっと人を楽しませる出題をする筈だ。　下手な出題では、売れぬ！
我ら物好きの研究対象にしかならぬ。

Mr. Hui の方法の特徴は　【関数 guess （推測）】である。
前報で、「簡易法として 関数 assign のみで可」と云う山下の説（p.13-14）もあったが、逆に中野は「関数 guess のみでも可」の例を示して置いたた（p.5）。

```
sdk4n =: 3 : 'guessa4n ^:(+/(0=y.)) y.'     NB. (n26)
```

５－ｂ　　山下報告　への　補足

１）　Clark 問題に関して云えば、Adrian Smith の紹介記事中の答（前節４．APL）は、この山下報告中には見当らない。　中野の estimation では clark 問題の複数解は 15220 通りもある（計算ミスがなければ）。
そして Clark の示した原解答は、その中で（ゼロ・オリジンで） 4857 番目に当たる。
かくて、西川が気にしていた Clark's Prob. は解けた。
もしも「それは、答が判っているから出来たのだろう」と申される方には、申し上げたい。　「解は　１５２２０　通りありますが、全部、示さにゃいけませんか？」と。

ついでに云えば、中野の phrase ga ^:(n) での単一解は最初（ゼロ・オリジンで０番目）に当たり、山下流の最終解の方は、15219 番目のものである。
解の全てを計算し尽くすまでに、中野のマシン（西川の最新機と同じく Sharpの PC-AL70F ）で　36 sec 掛った。

２）さらに調べる。　clark の原問題から given （指定）を１ケ減した問題 clark1 では、複数解数　２２７２７通り（演算 60 sec）であった。
さらに、指定を減らした（当初より２ケ減）clark2 では、複数解　７００２４通り（演算 180 sec）となる。
さらに、当初より３ケ減の clark3 の問題では、複数解はなんと、１６８９１９通り（演算 480 sec）、まさに「１６万超」である。　４ケ以上の減では、さすがにエラー out of memory で断念せざるを得なかった。

複数解の意味とその規模の理解は得られたであろうか？

３）上の逆に： given を追加してゆけば、解は絞り込まれる。

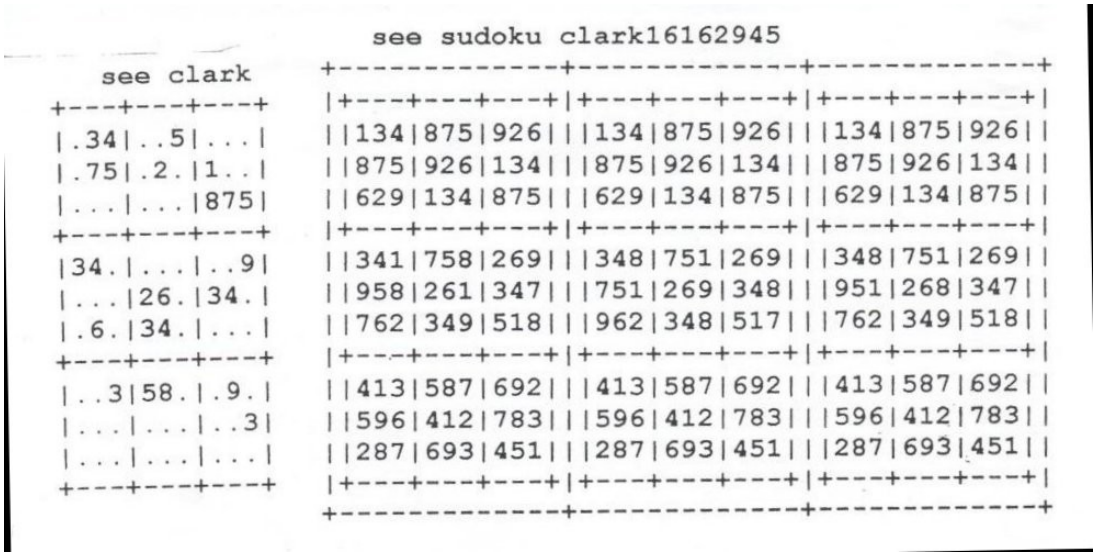
ａ．１ケ追加　：　原 clark に、０オリジンで (1,5)要素に 8 を追加の問題　clark8
　　では、ほぼ半減の ７５６４通り となる。　これは、あくまで、ほぼである。
もしも、異なる数 6 を追加した clark6 ならば、７６５６通りであった。
同程度だが、少しは異なる。

ｂ．２ケ追加　：　上の clark6 に、さらにもう １ケ 追加では、解は ４３７０通り。

ｃ．３ケ追加　：　上に、さらに１ケ、計３ケ追加では、解は　２３００通り。　同様に

ｄ．４ケ、５ケ、６ケ、７ケ、８ケ追加　では　３９５通り、　６３通り、　４２通り、
　　１２通り、　３通り。

下掲はその３通りの図。　左端は（８ケ追加指定された）問題、その右の３例が解。
さらに、３例中の中央が、英国の原著者 Clark　の　解答に相当する（文献７）。
つまり、原 clark 問題に、更に ９ケか１０ケ　追加指定し given を 35 ほどにすれば
良い。　前報（p.15）で、fill (given) を　25 から 40　としたのは近い。



4)　　山下の特別な関数　sdky の替わりに、単に phrase gz =: {: & g だけ
でも良い。　これは、guess を行った結果で, 毎回、その末尾を選択する意味だ。
これを　ga =: {. & g　とすれば、毎回、先頭を選択する。　いずれも、最終の解は
１つになるのは当然だ（毎回１つを選んでいるのだから）。　これらは８．節の末尾に
有効な phrase として追加しておく。　同じように、毎回 ２番目を選ぶとか、あるいは
毎回、末尾から２番目を選ぶように指定すること等も出来る。
こうした場合は、tacit な定義よりも sdky 的　な explicit な関数定義が楽である。
原理の説明に、以下に、 関数 guess の動きを示す。
材料に、全部が open な　問題　clarka0　から出発すると判りやすい。

```
   g clarka0
1 0 0 0 0 0 0 0 0 .........
2 0 0 0 0 0 0 0 0 .........
3 0 0 0 0 0 0 0 0 .........
4 0 0 0 0 0 0 0 0 .........
5 0 0 0 0 0 0 0 0 .........
6 0 0 0 0 0 0 0 0 .........
7 0 0 0 0 0 0 0 0 .........
```

8 0 0 0 0 0 0 0 0 .....
9 0 0 0 0 0 0 0 0 .....　**9**　行、　各行 **81** 文字を予測 **gurss** した。

もしも、関数　**ga** を用いるならば、上の先頭の　１行のみを選び、また、もしも **gz** ならば最下の１行のみ を選ぶ。　連続作業の例として

gz ^:(i.10) clarka0　　ならば、結果は
0 0 0 0 0 0 0 0 0 .....　　各行　**81**　文字
9 0 0 0 0 0 0 0 0 .....
9 8 0 0 0 0 0 0 0 0 .....
9 8 7 0 0 0 0 0 0 0 0 .....
9 8 7 6 0 0 0 0 0 0 0 .....
9 8 7 6 5 0 0 0 0 0 0 0 .....
9 8 7 6 5 4 0 0 0 0 0 0 0 .....
9 8 7 6 5 4 3 0 0 0 0 0 0 0 .....
9 8 7 6 5 4 3 2 0 0 0 0 0 0 0 .....
9 8 7 6 5 4 3 2 1 6 0 0 0 0 0 0 0 .....

これを　**gz ^:(1 + i.10) clarka0**　とすれば

9 0 0 0 0 0 0 0 0 .....

・・・・・・・・・・・・・・・

9 8 7 6 5 4 3 2 1 6 0 0 0 0 0 0 0 .....
9 8 7 6 5 4 3 2 1 6 5 0 0 0 0 0 0 0 .....

この流儀で　**n = 48** 回　**gz ^: (n) clarka0**　を反復すれば
単一解（先頭行は　**987654321**、次行は **654.....**）が得られる・・・類である。
同じ内容が、ただ今、山下ＦＡＸでもあった（文献８）。　同じく　４８回 だ。

印刷では多行になりかねないが、本来は　**81** 字で　１行である。

gz ^:(1)　末尾の１行（８１字）を１回選ぶ。　以下、その反復。
ga ^:(n) なら、毎回、先頭の１行８１字を選択する事を **n** 回 続ける。
次の候補が見つからなくなれば、終了するか、または **index error** などで打ち切り。
運よく、最後までラン出来れば、山下の云う最終単一解になるが、それは、今の例では
１万５千もの解の中の、意識的な １例に過ぎない事を理解しておくべきだ。
もっと正確には、「そんな問題を作る奴こそブタである！」。

５－ｃ　　**Adrian Smith** の記事は　ブタ

Vector誌に　**Mr. Hui** に続いて掲載されているが、**Hui** の３倍位、長大である。
しかし、原著でなく紹介記事の為か？判り難い。　またＡＰＬが、 **APL2** でなくて
Dyalog APLなので、不便である。　記事の後半（文献4、p.58）に、**Ellis Morgan** が
サイズ　**27 by 27 with cells and squares bordered** を取り上げた話がある。
いわゆる **BOX**（あるいは **BLOCK**）が 正方形の場合、「数独」問題のサイズ　**n** は
n = (2, 3, 4, 5, 6 ....) の　平方値な筈であるから、　**25** 次の次は **36**次である。
これは、いずれも、中野が問題作成例を示してある。
27次 では、 **BOX** は　矩形とならざるを得ない。　実例が示されないのは残念！

６．　候補選び、　　中野 の　Hybrid 法

２月報告「Ｊ の Labs システムによる　Hui のプログラムのトレース」（文献３）の執筆に、西川は精魂を使い果たしたらしい（文献１０）。
ただし、老友は Labs システムを知らぬので、口惜しい思いをさせられた。
判ったのは、関数 guess の基である候補者選び Candidates の力説であった。
中野は、 Mr. Hui の SUDOKU　論文到着（2005. Nov.28）以前の試みで、候補者選びを専らにしていたので、「やはりな」と思った。　関係事項を述べて見る。
さらに最近（2.14）のメール　JAPLA discussion (SHIMURA)（文献９）で気付いたが仮称 godspiral氏の Hui の改訂版論文に関しても、少し、いじって見たので、共に、コメントして置こう。

中野の　ｒｃｂ　法

生身の人間が sudoku 問題を解く操作、すなわち ｒ（横行）、ｃ（縦列）、ｂ（BOX） に着目して、重複なきよう、忠実に探索すると云う方法である。

古いものより、最新の例を、仮称 godspiral 氏（以下 god と呼ぼう）から採る。
実例には、以下の　3 box horizontal pattern　（god2 と名付けよう）

```
1 0 0  9 2 0  0 0 0
5 2 4  0 1 7  0 0 9
0 0 0  0 0 0  2 7 1
```

と

9 x 9　の問題　　x2　がある（見易く、ブランクを挿入）。

```
x2 =: |;._2 noun define
0 4 3  0 8 0  2 5 0
6 0 0  0 0 0  0 0 0
0 0 0  0 0 1  0 9 4
9 0 0  0 0 4  0 7 0
0 0 0  6 0 8  0 0 0
0 1 0  2 0 0  0 0 3
8 2 0  5 0 0  0 0 0
0 0 0  0 0 0  0 0 5
0 3 4  0 9 0  7 1 0
)
```

この問題 x2 は 、残念ながら、Hui の関数 sudoku でも、有効 phrase g の組合わせでも、全く歯が立たない。
また、 god2 は　3 x 9 の型で、元来、 hui の議論の外である。
（その後、山下が解いた天才的報告あり、それは後節で述べる。）

もっとも、超 Hui を目指した 中野 は、前報で　box が 矩形 の場合も扱った。
しかし、両者とも、いやらしい難問である（正に godspiral !）。
難問の理由は、空所への候補者が、全て　３組以上で、且つ、殆ど同じものであるので選択の手がかりが無い。　全く、盲滅法のトライアルを強いられるのだ。
それでも、何とかやった例を示す。

先ず　god2 の場合である。

```
初回原データ    god20 =. god2
        行列化    god2m =. 3 9 $ god20
解析データ
        行  r    r0=.   0{god2m
                  r1=.   1{god2m
                  r2=.   2{god2m
             BOX    b0=. ,(3{."1 god2m)
                  b1=. , _3{."1 (6 {. "1 god2m)
                  b2=. , _3{."1 bod2m
             列  c   c0=. 0{rgod2m =. |: god2m
                  c1=. 1{rgod2m
                  c2=. 2{rgod2m
                   ・ ・ ・ ・ ・ ・ ・ ・
                   ・ ・ ・ ・ ・ ・ ・ ・
                  c8=. 8{rgod2m
     データ行列
          n123 =.      r0, b0, c0,  r0, b0, c1,  r0, b0, c2
          n123 =. n123, r0, b1, c3,  r0, b1, c4,  r0, b1, c5
          n123 =. n123, r0, b2, c6,  r0, b2, c7,  r0, b2, c8
               ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
               ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
          n123 =. n123, r2, b2, c6,  r2, b2, c7,  r2, b2, c8
         n123m =. 27 21 $ n123

      候補探しプログラム     cand
          cand =: 3 : 0
          :
       NB. x.  n123  ,  y.  god20
              wr =: 1!:2&2
            i10 =: i.10
          im=. 0{ $ x.
           i =. 0
           while.   i<im  do.
              if. (i { y.)>0 do. goto_i. end.
              mi =. i { x.
              ans=. (-.(i10 e. mi))# i10
              wr 'i=   ', ": i
              wr  ans
           label_i.  i=.i+1
           end.
              ' end '
           )
```

　　これを、手直しして、 候補が（ブランクで無い）１ケ　の時は、その番号の位置に、候補値を代入する事も出来る。　中野は Hui 論文以前には、この方法を利用した。　２ケ以上の複数候補では、ランの途中で、ポーズを取り、任意に選んだ候補値をＫＢから、入力出来るようにもした。　機械と人間との ハイブリッド法である。
　" Hybrid SDK" 法とでも呼びたい。　元来「数独」の解法には、パソコン利用は「禁じ手」である。　消しゴムさえ使うな！　ひたすら黙視せよ。　鉛筆のみ可！

機械は「問題作成者」側が補助的に利用するものである。　従って、ハイブリッドは
正攻法と云えよう。

演算結果例

原データ　god20　1 0 0 9 2 0 0 0 0
5 2 4 0 1 7 0 0 9
0 0 0 0 0 0 2 7 1

1回後　god21　1 0 0 9 2 0 0 0 0
5 2 4 3 1 7 6 8 9
0 0 0 0 0 0 2 7 1

２回後　god22　1 0 0 9 2 0 3 4 5
5 2 4 3 1 7 6 8 9
0 0 0 0 0 0 2 7 1

３回後　god23　1 6 7 9 2 8 3 4 5
5 2 4 3 1 7 6 8 9
0 0 0　0 0 0　2 7 1

４回後　god24　1 6 7 9 2 8 3 4 5
5 2 4 3 1 7 6 8 9
3 8 9　4 5 6　2 7 1

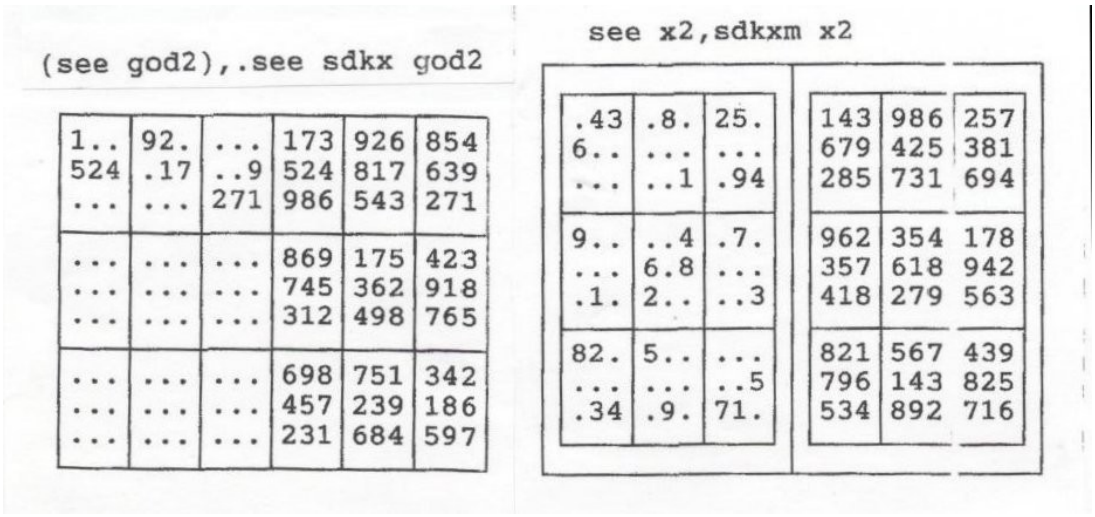
（完了）

中野の　Ｈｙｂｒｉｄ　法

前節の新情報　Mr. godspiral 提出の難問　x2 へのトライの例を示す。
サイズ 9x9 、given 26、 open 55 で難易度は 〜 medium 級だと思うが？
なんと、Mr. Hui の流儀では、何んとも、歯が立たぬ。
出題者のプログラムもまた【超 Hui】 を自称するだけあって、？？？
易経的プログラムの解釈はさておき、問題が解ければよかろう。　挑戦する。
原理は、候補者選びのＨｙｂｒｉｄ法だ。　時間が惜しいので結果のみを書こう。
ただ今、山下から、成功のＦＡＸがあったので、話はここで打ち切る。

７．　易経　と　山下解

話が終盤の去る２月１４日、重大なるメール "JAPLA discussion" が飛び込んで
来た。　件名は (SHIMURA) sudoku で、「易経」ファンはいらっしゃいませんか？
内容は、海外メール情報 「Hui の改良版」、発信者　<godspiral2000@yahoo.ca>
俗名は不明、カナダ発信と見えた。　内容は、e-mail 特有の「分かち書き」で、文章は
途中でズタズタ、従って、Ｊ の Script は　エラーの連続、何とか判読した一部を
稿末にて紹介します（参考になるか？）。
これに対し、１６日の西川メールは「マニアックな Ｊ のプログラムは Hui で沢山、
私はくたびれた。　老友は頑張っている。　若者に期待！」の意味であった。
頑張って！ は ＡＰＬ からみの Clark 問題を差すらしいが、これは　老友が解いた
後の話。　そこで、旧制・高等学校で、理系と謂えども、四書「大学、中庸・・」や
五経「易経・・春秋・」を経験した老人の出番となった。　例題に中野は悪戦苦闘。
山下はアッサリ解いた「何が難問なのか？　アッケにとられるよ！」　（文献１１）
成功の原因は、山下の天才的直感の他に、或いは、古いなじみの　Ｊシステム　JPCが
効果あったかも？　これは、西川が、旧版ユーザー用に、Hui の　SUDOKU Solver を、
わざわざ書き直したものである。　天才的な山下関数は「むすび」の直前に示した。



８．　【付録】　Ｈｕｉ　への　改良版プログラム（いわゆる易経）

from <godspiral2000@yahoo.ca>　to　<jcd02773@nifty.com>
via <jcd02773@nifty.ne.jp> and <JAPLA@aplsoft.co.jp>
彼のコメント文は適宜省略。　文番号（NB. 以下）「付き」は **Hui** からの借用、
「付かぬ」は **godspira**氏の改良部分。　中野の転写誤記あらば御容赦！

```
x2 =: ];._ noun define
043080250
600000000
000001094
900004070
000608000
010200003
820500000
000000005
034090710
)
fltr =: 1 : '(I. u. y.){y.'
x2=: ,,. dig2list(x2)
t =: x2
dig2list =: ,.&.":"0

list2dig =: 10 p.~|."1
grids =: _9{."_1 ((((0{$ grids)%10), 10)$ grids)
grids =: ,.&.":"(,.grids)

j     =. (]/. i.@#) ,{;~3#i.3        NB. (h1)
r     =. 9#i.9 9                     NB. (h2)
c     =. 81$|:i.9 9                  NB. (h3)
b     =. (,j{9#i.9) { j                        NB. (h4)
                                          NB. ここまでは　=.　を使用
I     =: ~."1 r,.c,.b                NB. (h5)
R     =: j,(,|:)i.9 9                NB. (h6)

regions =: R"_ {"_ 1 ]               NB. (h7)
```

```
   free    =: 0&= > (1+i.9)"_ e."1 I&{       NB. (h8)
   free2   =: avec@: free
   ok      =: (27 9$1)"_ -:"2 (0&= +. ~:"1)@ regions       NB. (h9)

   ac      =: +/ .*&(1+i.9) * 1: = +/"1      NB.  (h10)
   NB.  ac = ac1 * ac2
   ac1     =:  1=(+/"1)
   ac2     =:  +/ .*&(1+i.9)
   ac3     =:  3 : '+/"1 (1+i.9) * "1 y.'            NB. ac3 equiv to ac2.

   ar      =: 3 : 0                        NB.  (h11)
    m=. 1=+/"2 R{y.                        NB.  (h12)
    j=. I. +./"1 m                         NB.  (h13)
    k=. 1 i."1~ j{m                        NB.  (h14)
    i=. ,(k{"_1 |:"2 (j{R){y.) #"1 j{R             NB.  (h15)
    (1+k) i}81$0                           NB.  (h16)
   )                                 NB.  (h17)
   NB.  god2    100920000
   NB.          524017009
   NB.          000000271

b1b =: (3 3 3 $ (<"1 (_3+\,((18+i.9){R))))
b1  =: >(<<.(<<.(<"1(_3\ , 9{.R)),((, 0{"1 b1b),(,1{21 b1b),(,2{"1 b1b))))
b2 =: 3#(,.18 3 $ b1)
b2b =: (3 #(9{. R)),(3 #(9|.R))
b2 =: ;"1 b2b -. "1 b1
b3b =: (27 9 $ ,(3#(3 27 $ ,((9+i.9){ R)))),(27 9 $ ,((18+i.9){R))
b3  =: "1 b3b -. "1 b1

boxed =: b1; b2; b3
avecs =: 3 : 0
   b1t =: +./"2 (b1){y.
   b2t =: +./"2 (b2){y.
   mp=: b1t *. (-. b2t)
   fltb =: (+/"1 mp > 0)#]
  (((fltb b3){ y.)>"_2 1 (fltb mp))(fltb b3)} y.
)

fullass =: (+ ((ac1*ac3) >.ar)@ free2 "1)^:_ "1
assign   =: (+ ((ac) >. ar)@free2)^:_"1       NB. modify  (h18)

   guessa   =: 3 : 0                       NB.  (h19)
      if. -. 0 e. y. do. ,:y. return. end.     NB.  (h20)
      b=. free y.                          NB.  (h21)
      i=. (i.<./) (+/"1 b){10,}.i.10           NB.  (h22)
      y. +"1 (1+I.i{b)*/i=i.81                 NB.  (h23)
    )                                 NB.  (h24)

   guess    =: ; @: (<@guessa"1)             NB.  (h25)

   Fullsudoku =: guess @: (ok # ])@:fullass^:_@,"1
```

```
p96=:+/ listdig( 3{."1 Fullsudoku grids)

assign2 =: (+((ac) >. ar)@free)^:_"1
  sudoku   =: guess @: (ok # ]) @: assign ^:_ @ ,        NB.  (h26)
  see1     =: (;~9$1 0 0)&(<;.1) @ ({&'.123456789') @ (9 9&$) @ ,  NB.(h27)
  see      =: <@see1"1`see1@.(1:=#@$)       NB.  (h28)
  diff     =: * 0&=@}:@(0&,)              NB.  (h29)
  f =: (+(ac >. ar)@free)
  f1 =: +(ac1*ac3)@free
  f2=: (+ ar@free)
```

その他の追加資料として、 one step 推進用などの phrases が９ケほど。

```
(P1)  f=: + (ac >. ar)@free           NB. one step of assign
(P2)  see t=: f^:a: x                 NB. forced moves leading from grid x
(P3)  see diff t                      NB. differences from one grid to the next
(P4)  see assign  x                   NB. same as the last grid above
(P5)  see g=: guess (ok#|) assign x     NB. guesses after exausting forced move
(P6)  see t0=: f^:a: 0{g              NB. forced moves leading from guess 0
(P7)  see diff t0                     NB. differences from one grid to the next
(P8)  see t1=: f^:a: 1{g              NB. forced moves leading from guess 1
(P9)  see diff t1                     NB. differences from one grid to the next;
(n10)  ga =: {. & g ,   (n11)  gz =: {: & g        NB. NAKANO                  (y12)  sdky =:     {: @ g
NB.    YAMASHITA
```

追加　　山下　関数　　（易教対策に有効）

```
sdkx=:  ３: 0                    sdkxm=:  ３  : 0
   n =. 30                       n=. 1
  while. 50 >: n do.                while.  10>: n  do.
    z ＝.   sdky  ^: (n) y.              z=. g ^: (n)  y.
    if.  1=#z  do.  break.   end.        if.  1=#z  do.  break.   end.
    n=. n+10                        n=. n+2
  end.                            end.
    ’       ‘                                              ‘     ‘

    z                             z
  )                             )
```

９．　む　す　び

Sudoku（数独）の解法、 前報（その１）は 9 x 9 以外の　超 Hui を狙った。
本報（その２）は、残存問題　つまり、 Huiの流儀を深める狙いであったが、最後に
海外から　超 Hui とかの話題が飛び込んで来た。「易経」と「あだ名」されるほどに、
神秘的である。 でも、 9 x 9 の古典型の範囲内だった。　若者よ、脱皮させよ！
　しかし「数独」に、「根の公式」の如く、数値代入しての　直解は面白くない。
元来は「禁じ手」である。　やはり機械は、補助的に使うべきではないか？
人間と機械の混在する、ハイブリッドなやり方が望ましい。
マニアックな解釈問題に振り回された三老の感慨であろうか？

文　　献

１）中野・西川・山下：「数独とＪ」JAPLA 2006.Jan.28 報告資料
２）Roger Hui: "A Suduko Solver in J" VECTOR Vol.21 No.4 Autumn 2005 p.49～
表題の Suduko は、そのまま。　出版所のミスである。
－ａ）http://www.jsoftware.com/jwiki/Essays/Sudoku/
３）西川利男：「数独（SUDOKU）パズルを J で解く
－　Labs システムによる Hui のプログラムのトレース　－
JAPL研究会資料　2006/1/26　pp.13
４）Adrian Amith : " Sudoku with Dyalog APL from John Clark & Ellis Morgan"
VECTOR Vol.21 No.4 Autumn 2005 , pp.53-61
５）三枝協亮：「ＡＰＬ２学習支援パッケージ」 Kyosuke.saigusa@nifty.ne.jp
６）山下ＦＡＸ（2006.2.4,17:33） Clark 問題単一解
－ａ）（2006.2.12,17:33）「山下の　Clark 問題解法」 pp.4
７）中野ＦＡＸ（2006.2.7) :「APL の　Clark 問題解けた」　15000 -> 3 例
８）山下ＦＡＸ（2006.2.16,16:56） all blank , ４８回で完了
９）易経問題（志村）：from <godspiral2000@yahoo.ca> to <jcd02773@nifty.com>
１０）西川メール（2006.2.16, 8:23）(SHIMURA)sudoku - 西川です
１１）山下ＦＡＸ（2006.2.16,17:33）易経の問題、２題とも解けた。本当に難問なりや？

正　誤　表　（その１）　　p.13 西川旧版用プログラム

see1 =: (3 3 ,:3 3)&(<,.3)@ see0　最後の 3 の前に　. ドット を挿入