The chess knight's problems...

Stefan Behnel

August 2000

1 Trouble for the knight

The idea of looking at a single chess knight on the board goes all the way back to the 18th century, when Leonhard Euler [1], a famous swiss mathematician, introduced a problem during a stay in Berlin in 1758, which is usually referred to as *Springerproblem* or *The Knight's Tour*:

> Starting with an empty chess board, is there a path that has a knight visit all the fields (black and white) of the board exactly one time?

The knight is paticularly interesting because of his strange way of moving. It would be much less interesting to try that with the queen, for example, since most attempts should lead to a solution. The knight in turn has a very limited way of moving that keeps the number of accessible fields per move below nine.

The knight moves over the chess board in the way that is displayed here. So if the knight happens to be on the field marked with an S, he can access any of the X-fields in the next step.

0	X	0	X	0
X	0	0	0	X
0	0	S	0	0
X	0	0	0	X
0	X	0	X	0

When Euler first thought of that problem, he imagined an $8x8$ board, the regular chess board, while today's mathematical and computerized methods make us want to take a look at bigger boards, too.
But in order to find out about the chances of solving the problem, we have to take a closer look at it first.

2 Different flavours...

During the last almost 250 years, different ways of looking at the problem have evolved. It is remarkable, that even the smallest one among them took about 230 years to be sufficiently solved in a way that is scalable even for huge boards.
The problem of the Knight's Tours is often used as an example in graph theory, since it is easy to imagine and to try out but nevertheless a very big problem with many different flavours to emphasize on. It can be used to show the idea of symmetry, of directed and indirected tours in graphs and of paths and closed tours. Though it is a Hamiltonian problem, solutions can easily be found.
Since it was proven to have solutions for all boards $>= 5x5$ in the early 1990s, it has now become even more popular and commenly known and there were several attempts and applications in this field that reach from simple Java-Applets for demonstration and visualisation purposes up to parallel computation and neural networks (why ever...)

2.1 All under a general headline

The *Knight's Tour* represents a special case of one of the biggest problems in computer science: The *Hamilton Cycle*.
R.W. Hamilton was an irish (cheerz luds!) mathematician of the 19th century. The problem of the Hamilton Cycle describes the search for a path in a graph to connect all knodes so that they build a cycle, similar to "connect the dots" (Malen-nach-Zahlen) adding the slight difficulty that the numbers are missing and you have to find the right order yourself - and usually there are quite a few dots to connect...
The Hamilton Cycles belong to the class of NP-complete problems and are therefore believed to be only solvable with an afford that grows exponentially with the size of the problem.

To find a Knight's Tour is by far more simple since the limitation to a chess board provides some benefits like indirectness, symmetry and - here we go: - recursion. In fact, if recursion is applied to the board, finding a Knight's Tour becomes so "easy" that todays home computers do this for giant boards with some billions of fields in an almost unmeasurably short period of time. You can just wait for it...

2.2 Finding a Knight's Tour

The problem of finding a single solution for the *Knight's Tour* was solved in the early 1990s by a group of students as a project for the german scientific contest "Jugend forscht" [2], [3]. Their algorithm finds a single solution on

a chess board of any size (>= $5x5$) within an almost unmeasurably short period of time.
The basic idea is quite simply generic recursion. They cut the big boards into very small ones ($5x5$, $6x6$, $6x7$, $6x8$, $7x7$, $7x8$, $8x8$), for which a solution can easily be found, and then reconnected the small boards to fill the bigger (and really big) boards. That way they only had to find solutions for small boards that match certain criteria, like a given starting and ending position (easy enough) that allowed the knight to move from the end position on one board to the starting point of the next one.
Using this algorithm, they could prove that any board bigger than or equal to 5x5 has at least one solution, as all boards with longer sides can be split into boards with the sizes mentioned above (length $N = a * 6 + b * 7 + c * 8$ for all $N > 17$, $N <= 17$ works anyway).
Now some of those solutions can be found in no-time. Normally there are quite a few to be found, so it's not that hard to find at least one.

Since this algorithm can be used for parallel computation, it is impossible (and unneccessary, BTW) to find a faster one.

2.3 Coming back to where we started

A variation of this problem makes the knight return to the starting point on the board. It's not really more afford to find a special solution like that since it only constraints the algorithm to end up on one of a certain number of fields, which is always possible if the knight actually *can* end up on those fields. The exceptions are given by the following idea:
A $5x5$ board has 25 fields. On each move, the color of the field the knight is standing on changes because of his way of moving. So if the first field was white, the second one will be black and so will be every field the knight accesses with an even number. Therefore the 25th field, the last one he moves on, will be white again. On this board, the first position and the last one always have the same color, so there is no way for the knight to move from the last one directly back to the starting position. Therefore re-entrant paths (real cycles) are only possible on boards with an even number of fields.

All boards (> $5x5$) with an even number of fields provide at least one solution for a cycle. Boards of sizes $3xN$ or $4xN$ may vary.

2.4 The Knight's Paths

Well, I didn't like all those *at least*s when I wrote this text, since it means that we do not actually know the exact number of solutions that can be found for a given board.
Now we are coming to the biggest problem of this little family: We are

trying to count the number of paths the knight can move along in order to access every field of the board exactly one time. This really involves finding all solutions and stupidly counting them. To get an idea of the size of the problem, you may take a look at the following table, that gives the number of valid board configurations on different squared chess boards.

Boardsize	Solutions	# valid	# max.
$4x4$	0	29,976	$1.05 * 10^5$
$5x5$	1,728	38,010,697	$2.79 * 10^{11}$
$6x6$	6,637,920	?	$4.73 * 10^{19}$
$7x7$	?	?	$5.15 * 10^{29}$
$8x8$	$8.12 * 10^{18}$	?	$3.58 * 10^{41}$

$8x8$ was found to have 8,121,130,233,753,702,400 directed paths by Löbbing and Wegener in 1996 [6].

2.5 Heuristics and estimates

2.5.1 The size of the problem

The maximal number of field configurations given above allows to estimate the size of the problem since those have to be tested in order to find all solutions. This number is nevertheless much bigger than the number of allowed paths since the knight normally runs into dead ends long before he can actually finish his journey across the board. The number is easily calculated since we can count the number of accessible fields for each position on the board and multiply them. This results in the maximal number of paths that may allow the knight to access all fields, but still includes everything that whould lead to a dead end situation.
An example: The knight starts in the top-left corner of the chess board. From that position he can access two other fields, making it two possible paths. Selecting one of them the knight now can access five new fields and so on. To find the maximal number of paths we can now multply the number of accessible fields on all his ways: $2 * 5 * \ldots$
Since the knight always has to access each field once on each path, this is just the same as if we multiplied the number of successors for each field on the board. We may argue that we have to decrease all factors (but the first one) by 1 since the knight always comes from one of those fields,so there'salways at least one less accessible, but since we search the solutions beginning on any of the fields on the board, any of them can be the first one. So we really end up multiplying all counters (believe me!).

Examples for the successor counter on $5x5$ and $6x6$ are given below.

Some test calculations show an average depth for backtracking at about

Table 1: Number of successors for $5x5$

2	3	4	3	2
3	4	6	4	3
4	6	8	6	4
3	4	6	4	3
2	3	4	3	2

Table 2: Number of successors for $6x6$

2	3	4	4	3	2
3	4	6	6	4	3
4	6	8	8	6	4
4	6	8	8	6	4
3	4	6	6	4	3
2	3	4	4	3	2

two third of the way (60-70%) if no optimizations are used. This leads to the following calculation:

The maximal branching factor of the problem (i.e. the maximal number of successors that have to be tested in the next iteration) is eight, since one field has at most eight successors. As it can be seen from the boards above, if the board grows, the middle part will fill with a successor count of eight while the two rows and columns on both sides stay basically the same. Since the middle part is a square, it will grow much faster than the borders that grow linearly, so for big boards, the branching factor will grow against 8. That's why counting the Knight's Tours suddenly becomes such a big problem even if the board is only slightly growing. For now, we will just write b for this branching factor.
Now we see that there are $N*M$ valid configurations in the first iteration of an NxM board. There are about $(N*M)*b$ in the second, then $(N*M)*b*b$ and so on. That makes it $(N*M)*b^{N*M-1}$ in the last iteration. The resulting exponential function is

$$c(i) = (N*M)*b^{i-1}.$$

We see that by far most of the possible configurations reside at the end of the paths rather than at the beginning, simply because of the high branching factor, and so we can integrate the function to see how many of them

do. The integral of the function c is given by

$$I(i) = \int c(i) = \frac{N*M}{ln(b)} * b^{i-1}.$$

We already said that after about two third of the path the knight has to turn back and do backtracking. For generalisation purposes I will name this ratio δ in the following text. So now we take the integral on this interval to have an approximation for the valid board configurations:

$$P_{tried} = I(\delta * N * M) - I(1) = \frac{N*M}{ln(b)} * (b^{\delta*N*M-1} - 1)$$

and the integral of the interval that represents the invalid configurations (since the knight had to return earlier on) is given by:

$$P_{backtracked} = I(N * M) - I(r * N * M) = \frac{N*M}{ln(b)} * (b^{N*M-1} - b^{\delta*N*M-1})$$

We now can calculate the relation between the two:

$$\frac{P_{tried}}{P_{backtracked}} = \frac{b^{\delta*N*M-1}-1}{b^{N*M-1}-b^{\delta*N*M-1}} \approx \frac{b^{\delta*N*M-1}}{b^{N*M-1}-b^{\delta*N*M-1}}$$

This is a constant for a given board. It gives us an approximate relation between the maximum number of board configurations and the number of valid ones and allows us to estimate the size of the problem more exactly.

2.5.2 The average branching factor

The average branching factor is easily calculated. It is the quotient of the sum of all field successor counters on the board. We will assume the board to have NxM fields with $N, M > 5$. As it can be seen from the examples 1 and 2 above, the successor counters change very regularly while the board grows, so we will now develop a term for their sum.

Table 3: Number of successors for $6x6$

2	3	4	4	3	2
3	4	6	6	4	3
4	6	8	8	6	4
4	6	8	8	6	4
3	4	6	6	4	3
2	3	4	4	3	2

The four fields on the little square in the corners (showing the numbers 2,3 and 4) always stay the same, so now we count the fields showing a two or a three plus the four fields showing a four, which gives us the constant $4*2+8*3+4*4$ for our term.
As the field grows, the number of fields at the borders that have a four on them grows with it. The first and last rows and columns will be filled with fours, except for the fields in the corner, so this leads to the terms $2(N-4)*4+2(M-4)*4$. The same situation occurs for the fields with a six on them: $2(N-4)*6+2(M-4)*6$. The rest of the board (a squared piece in the middle) will be filled up by a successor counter of eight. This piece is always four rows and columns smaller than the entire board, so the term for this part is $(N-4)(M-4)*8$.

Therefore our sum becomes

$$\begin{aligned}\sum &= 4*2+8*3+4*4+2(N-4)*4+2(M-4)*4+ \\ &\quad +2(N-4)*6+2(M-4)*6+(N-4)(M-4)*8 \\ &= 8*N*M-12*N-12*M+16\end{aligned}$$

Now we can calculate the average branching factor as

$$b=\frac{8*N*M-12*N-12*M+16}{N*M}$$

In the special case of a squared board this simplifies into

$$b=\frac{8*N^2-24*N+16}{N^2}$$

The form $b=8-\frac{12}{M}-\frac{12}{N}+\frac{16}{N*M}$ allows us to see that the branching factor quickly converges against 8 if the board grows. That means that the size of the problem grows even faster the bigger the board becomes. On a $24x24$ board, the average branching factor already is bigger than 7.

2.5.3 It's BIG !

Putting it all together, we now have found a formula that allowes us to estimate the size of the problem on a given chess board of NxM fields in terms of the number of possible field configurations that could be tested.

$$\mathcal{S}=\frac{b^{\delta*N*M-1}}{b^{N*M-1}-b^{\delta*N*M-1}}$$

$$\text{with } b=\frac{8*N*M-12*N-12*M+16}{N*M}$$

So now here is a new table which gives the new approximation of the already mentioned boards considering an average depth for backtracking between 62% and 68%.

Size	avg. bf	known valid	min. (62%)	max. (68%)	absolute max.
$4x4$	3.00	$2.99 * 10^4$	$1.57 * 10^3$	$3.10 * 10^3$	$1.05 * 10^5$
$5x5$	3.84	$3.80 * 10^7$	$4.43 * 10^6$	$2.53 * 10^7$	$2.79 * 10^{11}$
$6x6$	4.44	?	$2.75 * 10^{11}$	$5.49 * 10^{12}$	$4.73 * 10^{19}$
$7x7$	4.90	?	$2.64 * 10^{17}$	$2.30 * 10^{19}$	$5.15 * 10^{29}$
$8x8$	5.25	?	$3.59 * 10^{24}$	$1.73 * 10^{27}$	$3.58 * 10^{41}$

As I mentioned above, the constant δ is the only weak part of this formula as it has a relatively high degree of freedom. To find out in about which depth backtracking usually happens, the problem would have to be calculated.
But the benefit of this formula is that it also works when optimizations are used, so it allows to estimate the problem size during the search for solutions. We only have to keep a counter for the number of moves that were tried so far and another one for the sum of the depths in which the knight had to turn back for whatever reason. The quotient of both gives us the average depth of backtracking that, divided by the number of fields, results in a value for δ.
This allows us to predict the remaining number of moves to be tested and even the time this may take as we can easily calculate the number of moves made per second so far.

2.6 Symmetry

Since symmetry tends to decrease the number of field configurations that have to be tested by factors, it is usually one of the first ideas to implement. The following symmetries exist on any board:

1	1	3	1	1
1	1	3	1	1
2	2	4	2	2
1	1	3	1	1
1	1	3	1	1

Different areas with the same numbers are symmetric and therefore provide the same number of solutions. The areas marked with *2* only exist on boards with an odd height, the areas *3* only on boards with odd width. If both width and height are odd, area *4* has to be taken into account, too. On evenly sided boards only the areas marked with*1* exist and only one of them has to be tested.
On square boards, the areas *2* and *3* provide the same number of solutions.

Often enough other symmetries can be found, especially if the board is squared. When the knight starts on one of the diagonals of a squared

board, it doesn't really matter into which half of the board he moves first, the number of solutions will be the same.

References

[1] Leonhard Euler, *Mémoires de Berlin: 1759*, 1766

[2] Tanja Hinrichs, Hussein Morsy, Axel Conrad, Ingo Wegener, *Knight's Tour - Eine kurze Reise durch die Welt des Springers*, 1999, http://www.axel-conrad.de/springer/springer.html

[3] Tanja Hinrichs, Hussein Morsy, Axel Conrad, Ingo Wegener, *Solution of the Knight's Hamiltonian Path Problem on Chessboards*, 1994, Discrete Applied Mathematics 50, p. 125-134

[4] A. Roth, *The Problem of the Knight - A Fast and Simple Algorithm*, http://www.mathsource.com/cgi-bin/msitem?0202-127

[5] Ian Parberry, *An Efficient Algorithm for the Knight's Tour Problem*, 1997, Discrete Applied Mathematics 73, p. 251-260, http://mycroft.csci.unt.edu/~ian/papers/knight2.html

[6] I. Wegener, M. Löbbing, *The Number of Knight's Tours Equals 33,439,123,484,294 - Counting with Binary Decision Diagrams*, 1996, Electronic J. Combinatorics 3, R5 1-4, http://www.combinatorics.org/Volume_3/volume3.html#R5

[7] Eric W. Weisstein, Wolfram Research, Inc., *Knight's Tour*, 1996-2000, http://mathworld.wolfram.com/KnightsTour.html

[8] Stefan Behnel, *Knight's Tour - a cluster application in Java*, 1998, http://www.sc.cs.tu-bs.de/pare/projekte/behnel/index.html

[9] Stefan Behnel, *Springer-Problem '97 for Win32*, 1997, http://www.tu-bs.de/~y0007726/projekte/springer.zip

[10] Stefan Behnel, *Problems for the chess knight*, 2000, http://www.behnel.de/knight.html