Pattern Tours, Part 2: Tour Basics

Perspective

As described in the first article in this series [1], a pattern can be constructed from a tour that visits every cell in some order and colors them according to black-and-white band. Thus, a tour serves to distribute the colors of the band over all cells in a grid.

The number of possible tours for all but trivially small grids is enormous. If there are k cells, there are k possibilities for the first cell on the tour. This eliminates one cell but leaves $k-1$ possibilities for the second cell. Therefore the number of possible tours for a grid of k cells is $k \times (k-1) \times (k-2) \times \ldots 3 \times 2 \times 1 = k!$ (k factorial). For an 8×8 grid, there are 64 cells and the number of possible tours is 64!, which is

126,886,932,185,884,164,103,433,389,335,161,
480,802,865,516,174,545,192,198,801,894,375,
214,704,230,400,000,000,000,000

Obviously, tours must be chosen with some plan or concept in mind, for a tour is the geometry from which the eventual pattern is crafted. A random tour is extremely unlikely to produce attractive results.

The problem of tour design is complicated by the fact that it alone does not determine the pattern; the band plays a strong role in the final result. This makes tour design challenging and interesting.

There are certain things that can be looked for in tours. One is some kind of pattern in the tour itself. A jumbled, chaotic tour, even if far from random, is unlikely to produce attractive results *unless* the band is developed along with the tour. This is possible — take any pattern and any tour, however disorganized, and read out the band. This band, when read in by the tour, will reproduce the original pattern. Some uses for tours and bands produced in this way will be explored in a later article, but the first concern is designing tours independently of bands.

While designing tours and bands independently requires both the application of some principles and some serendipity, it is the core of a process for getting attractive and unusual patterns.

If tours are taken alone, their design needs to be guided so that the tours themselves are attractive and interesting.

Tour Design

There are several properties to keep in mind when designing tours.

Symmetry: Symmetric designs are attractive to human beings (although no one knows exactly why). Various kinds of symmetries are possible for tours.

Repetition: Repetition of a unit within a design can be useful in tours just as it is in tilings, weaves, and other kinds of artistic constructions.

Variety: A little variety or an element of surprise can break an otherwise monotonous design and be aesthetically pleasing.

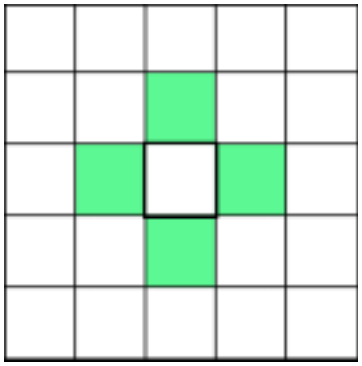
Continuity: Since a tour is a path in the general sense, there is some value in continuity. For example, the locations on a tour may be adjacent (their cells sharing a side). There are various kinds of continuity and technical names for them. The problem will be discussed in the context of constraints.

Tour Constraints

Constraints can be useful in design; they limit what is possible in a systematic way that prevents accidental aberrations, and they provide for a certain degree of regularity.

The kind of constraints here are local ones that allow tours to be built step-by-step. Such constraints might limit the distance from one location on the tour to the next or limit the possible directions in which the next location may lie.

The neighborhood concept from cellular automata [2] is a useful model for this kind of constraint. For example, the von Neumann neighborhood looks like this:



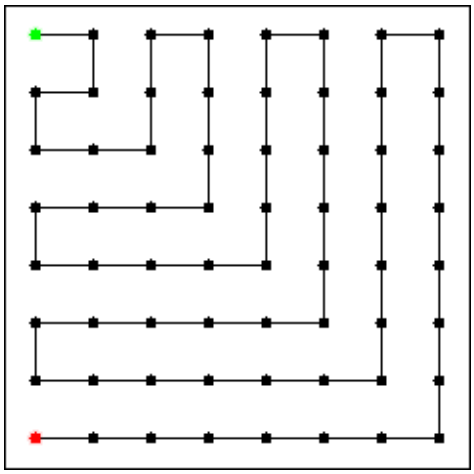
Viewed as a constraint, this neighborhood

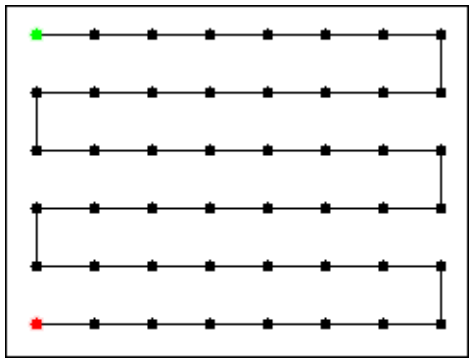
December 5, 2002; last revised December 15, 2004

requires the location following the central location to be one cell away, horizontally or vertically. Staying put is not an option, since a location cannot appear twice in a tour. Locations that are off the grid, when a cell is at an edge, obviously are excluded. Similarly, locations that already are on the tour are forbidden.

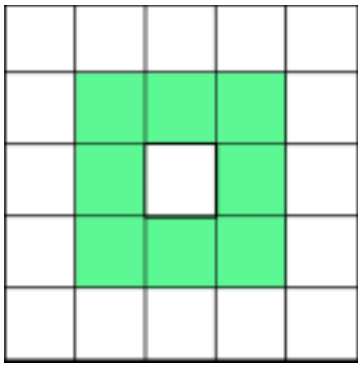
We'll call a tour constructed using this neighborhood a *von Neumann tour*. Von Neumann tours have a special kind of continuity, called *unicursal* in graph theory. Von Neumann tours also are *planar*, meaning there are no crossings on a line drawn along the tour.

Here are examples of von Neumann tours:

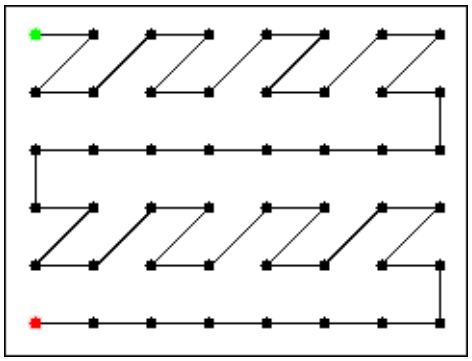


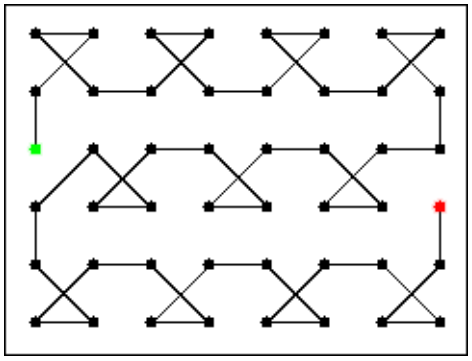


The Moore neighborhood allows more freedom of movement:

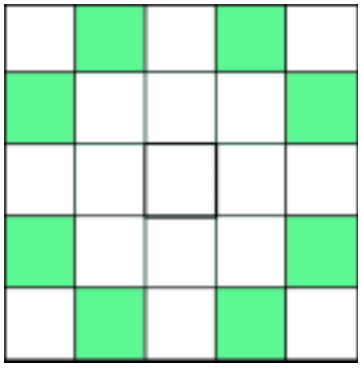


Moore tours include von Neumann tours as a subset, but they allow considerably more variety, including diagonal moves and non-planar tours. Here are some examples of Moore tours that are not von Neumann tours:





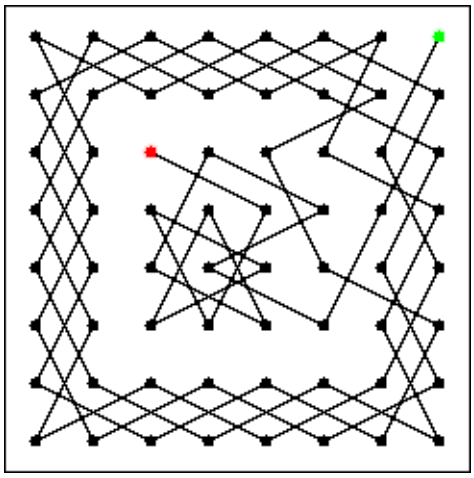
The legal moves of chess pieces can be described as neighborhoods. For example, the knight has the neighborhood



December 5, 2002; last revised December 15, 2004

Note that the knight "jumps"; it cannot move to an adjacent cell.

Knight's tours are sufficiently interesting and difficult to design that they have occupied the attention of chess players and mathematicians for centuries. We'll have more to say about knight's tours in another article. For now, here is an example:

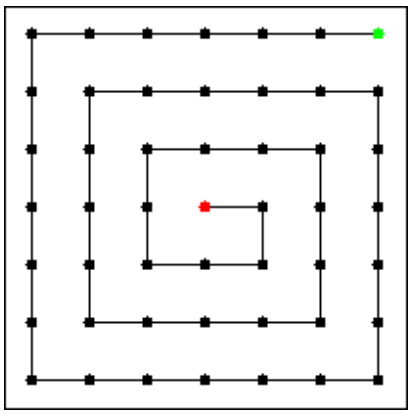


Neighborhoods provide constraints that limit the nature of tours. In constructing tours using neighborhoods, there are, of course, choices. One thing that can happen is getting into a situation in which no further move is possible. There are ways of dealing with this problem, which we'll discuss in a later article.

Tour Classification

Tours can be classified roughly according to the methods by which they can be constructed and places they can be found. In many cases, a tour will fit into more than one category, depending on how it is created or viewed.

Algorithmic Tours: These tours are constructed according to a fixed set of rules applied in a well-defined fashion. An example of an algorithmic tour is a square spiral:

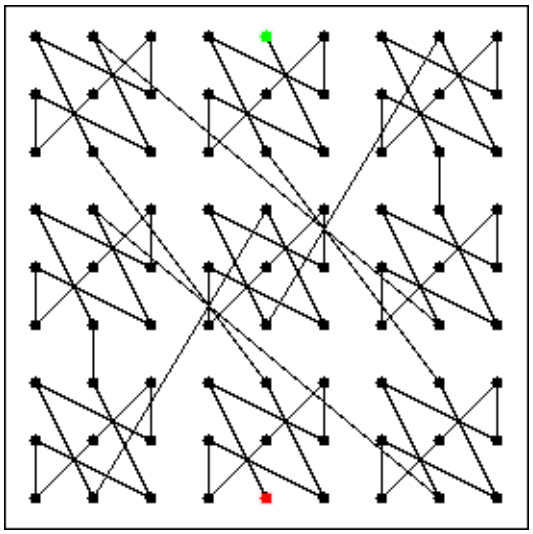


Note that this also is a von Neumann tour.

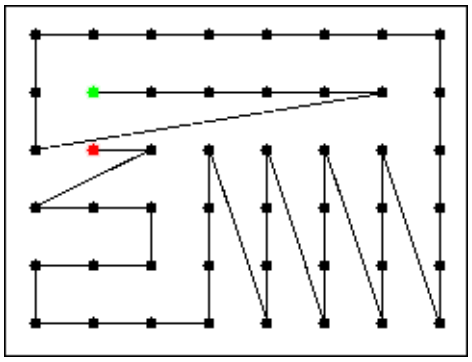
The geometry of this kind of tour is obvious, as are possible variations on it. You might find it illuminating to construct an algorithm that produces square-spiral tours.

Neighborhood-Constrained Tours: These tours are discussed above. They come in great variety and will be the subject of subsequent articles.

Numerically Derived Tours: These tours are derived from numerical problems and puzzles that are not directly related to tours but that nonetheless can be interpreted as tours. An example is this tour derived from a magic square, in which the sums of the rows, columns, and main diagonals are all equal:



Miscellaneous: There is the inevitable "other" category containing tours that do not fit elsewhere. Here is an example of a tour that was constructed by hand:

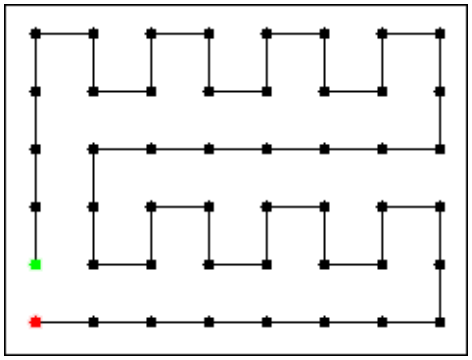


More Terminology

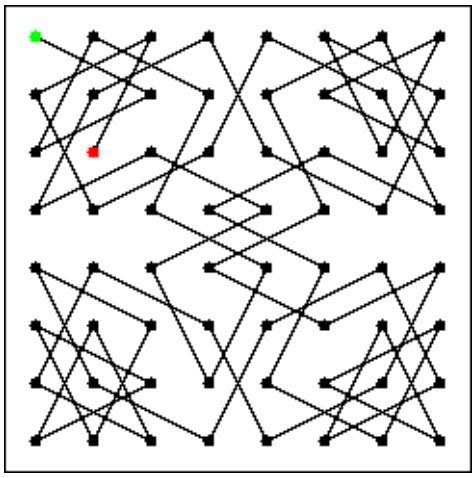
There are a few other terms related to tours that are important in some contexts:

Re-Entrant Tours: A tour whose last location is

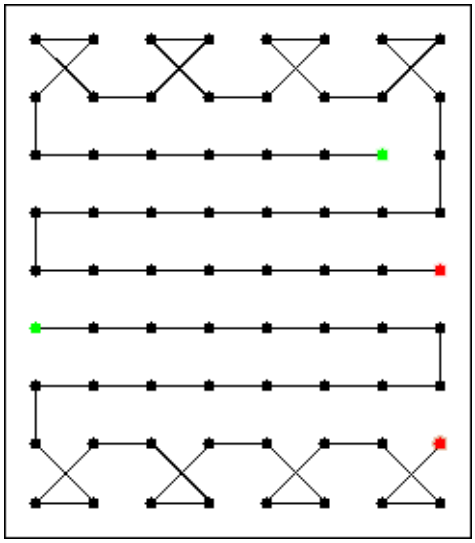
within a legal move of its first location is called *re-entrant*. An example of a re-entrant von Neumann tour is:



Here is an example of a re-entrant knight's tour:



Piece-Wise Tours: A piece-wise tour is one composed of parts that satisfy some condition, but the whole does not. Here is an example of a piece-wise Moore tour:



Such tours can be made into regular tours by connecting the end of one piece to the beginning of the other, but it often is more useful to view the parts independently.

Incomplete Tours: A path that does not include every location in a grid is called *incomplete* or *open*. Incomplete tours can be used for components of piece-wise tours or alone in some applications.

Overtours: A path that includes a location more that once is called an overtour. Overtours also have their applications, which will be discussed in a future article.

References:

1. *Pattern Tours, Part 1: Basic Concepts*, Ralph E. Griswold, 2004:
   http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_dt1.pdf
2. *Pattern Automata, Part 1: Basic Concepts*, Ralph E. Griswold, 2004:
   http://www.cs.arizona.edu/patterns/weaving/webdocs/gre_dda1.pdf

Ralph E. Griswold
Department of Computer Science
The University of Arizona
Tucson, Arizona