Every House on the Block: A generalized solution to creating ISR instances of large plan length

Remo Christen[1], Salomé Eriksson[1], Michael Katz[2], Emil Keyder[3], Christian Muise[4], Alice Petrov[4], Florian Pommerening[1], Jendrik Seipp[5], Silvan Sievers[1], and David Speck[6]

[1]University of Basel
[2]IBM T.J. Watson Research Center
[3]Invitae
[4]Queen's University
[5]Linköping University
[6]University of Freiburg

1 Introduction

Several strategies were tried throughout the course of the contest. From hand-crafted examples to exhaustive enumeration of smaller graphs with the aid of SAT solvers and knowledge compilers. Ultimately, a useful pattern for repeated movements was discovered. Since it is readily scalable, this is what is used for the submission. Throughout the remainder of the document, we describe the core concepts that make up the graph.

2 The "House" Widget

In order to encode bit flips in a graph, we leverage a five node subgraph we call the "house widget": a 4-cycle with two adjacent nodes leading to a 5th (essentially, a triangle sitting on top of a square). The house widget has a number of properties that make it ideal to use as a building block in creating exponential sequences.

1. The graph has an optimal "long" shortest reconfiguration sequence for ISR instances of order 5.
2. You can only place two tokens on this widget, meaning each step of the reconfiguration sequence consists of a maximum independent set. In other words, the sequence is "tight" and no additional nodes can be added to the independent set at any point.

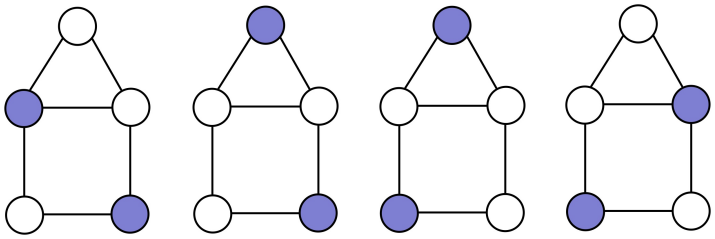

Figure 1: Reconfiguration sequence from "off" to "on"

3. The topmost node, which we call the "anchor", is occupied throughout the entire sequence with the exception of the starting state and ending state, and is required to switch the corners that the two tokens are on.

4. The sequence is unique. Thus, the solution space is a path and the behaviour of the widget is predictable.

In summary, we have a widget that remembers its setting, takes five nodes to do so, and three moves to make it happen. We call one setting "on" and the other setting "off" (they're symmetric).

3 Connecting Anchors

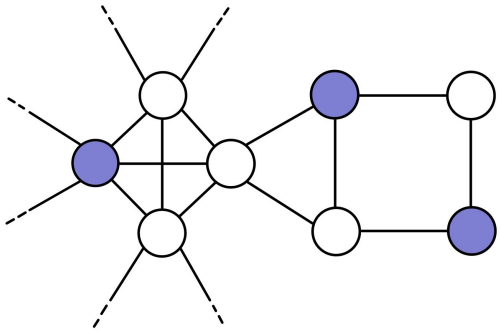

Figure 2: Four connected anchors; clearly, only one house can "flip" at a time

Since we treat our house widgets as bit flips, the first step in establishing an optimally long sequence is ensuring at most one house can be switching states at any given time. This is done by fully connecting the anchors of all houses in our graph. Recall that the anchor is both required to switch a house from "on"

to "off" and is occupied throughout the sequence. By making the anchors a fully connected subgraph, we guarantee no houses switch states simultaneously.

4 Flipping Bits

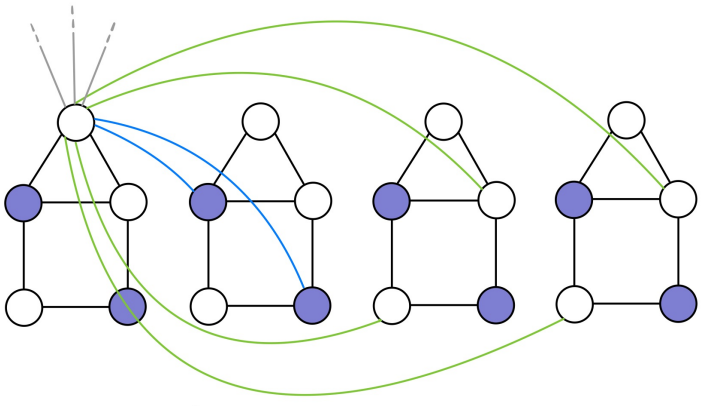

Figure 3: Edges for a single house. Blue and green edges correspond to Rules 1 and 2 respectively. Grey edges correspond to connections with other anchors.

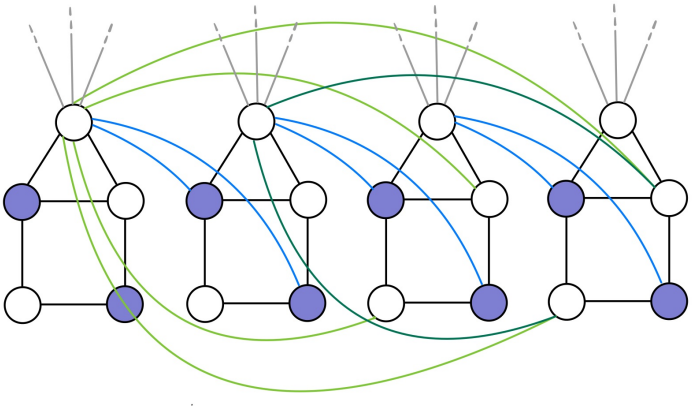

Figure 4: Four connected houses. Blue and green edges correspond to Rules 1 and 2 respectively. Assume all anchors are fully connected.

Now that we have a rigid set of widgets (referred to henceforth as bits) which we can turn "on" and "off", we must connect them in a manner that results in exponential growth. Assume bits are ordered (so there is a 1st bit, 2nd bit,

etc..). The key challenge is that any edge between house bases will permanently rule out a pair of bit configurations, thus all additional inter-house edges are between an anchor and bit values of another house. The order of bit flips is then enforced by the following relations in order to flip bit k.

Rule 1: Bit $k+1$ must be "on".

Rule 2: Bits $k+2 \cdots n$ must all be "off".

This sequence results in an exponential cost flipping things back and forth in order to get a low bit flipped.

5 Generating Graphs

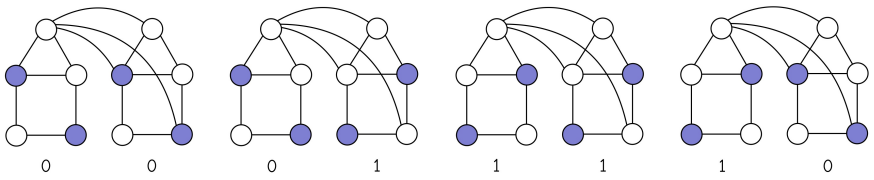

Figure 5: A series of bit flips for a graph of order 10. Note that each bit flip takes 3 moves, so the entire sequence is of length 9.

Putting everything together, graphs are generated as follows:

1. Create k "houses"
2. Make the anchors of each house a fully connected subgraph of order k
3. Add edges corresponding to the three bit flipping rules listed in Section 4, *Flipping Bits*

6 Adding Widgets

The final step is the strategy for adding an extra house. Suppose we have a sequence of k houses generated via the strategy above. We add house k+1 according to the following:

1. We can only flip house k+1 when the goal of house k is satisfied.
2. The new goal is the initial state of the k house sequence, combined with the flip of house k+1

This forces the plan length to double with each new house: achieve the old goal of the k house sequence, flip house k+1, go back to the initial state of the k house sequence.

7 Contest Instances

Since each size was a multiple of 5, the submissions for the graph track take 2, 10, and 20 houses respectively for graphs of order n=10, 50, and 100.

The lengths of the optimal plans found are as follows:

n	Length
10	9
50	3,069
100	3,145,725