A review on current HC problem solvers

1. Introduction

This documents aims to review the current state of the art methods to solve the HC problem. The methods discussed here are:

- Snakes and Ladders Heuristic
-

2. Snakes and Ladders

Worst-case algorithmic complexity In the algorithm implementation above, stage 3 has the largest search space and the stopping condition with the largest bound, so in the worst case dominates the execution time. In each iteration of stage 3 we allow as many opening transformations as are necessary to enable a floating transformation to close a gap. Then, there will be a sequence of O(n) such floating transformations that close gaps. If k is the maximum degree of the graph, then there are O(nk4) potential floating transformations at each step of the sequence. This number of transformations arises because the SLH floating 5-flo transformation requires four edges to be chosen, the first emanating from any vertex next to a gap, and each subsequent edge emanating from a vertex determined by the previous edge. Thus, there could be at most n gaps, and k edges from each 12 P. Baniasadi, V. Ejov, J.A. Filar, M. Haythorpe, S. Rossomakhine vertex. Then, to determine if such a transformation produces an ordering not already in the ordering list, we need to search the latter. There could be up to n3 orderings stored, in an index set, so using a binary search over the set is an O(log(n3)) = O(log(n)) process, and comparing two orderings is a O(n) process. There are, at most, n3 such iterations in stage 3. Theoretically, stages 1–3 could be repeated n times as the requirement to return from stage 3.3 to stage 1 is only that the number of gaps (at most n) has decreased by at least 1. Therefore the worst-case complexity of the algorithm detailed above is O(n × n3 × n × log(n) × nk4 × n) = O(n7 log(n)k4). For a sparse graph, where k is O(1), this reduces to O(n7 log(n)). Other operations (such as adding the newly constructed ordering to the ordering list) are dominated by the time taken to search for the transformations in stage 3. Our experience indicates, however, that this worst-case complexity is very unlikely to be encoun- tered in practice. Experimentally, we have seen that when SLH reaches stage 3 the number of gaps is very close to the minimal possible number, so stages 1–3 only need to be repeated a handful of times. Also, the likelihood of needing to open many gaps before one can be closed in stage 3 is extremely small. So, even in cases where a non-Hamiltonian graph is submitted to SLH the performance is likely to be closer to O(n5 log(n)k4), or O(n5 log(n)) for a sparse non-Hamiltonian graph. Furthermore, in our experiments we have seen that almost all Hamiltonian graphs are solved without ever needing to reach stage 2. In such cases the heuristic has complexity O(n3 + n2k4), or O(n3) for sparse Hamiltonian graphs. This bound arises because there are at most n2 iterations in stage 1,

at each iteration we search from a prescribed gap so there are O(k4) choices of floating transformations, and after each iteration we need to record the new ordering which is an O(n) proces