Experiment Descriptions

1 Definitions

Given a state space topology $\langle S, h \rangle$ with state space S and heuristic h:

Definition 1.1. Heuristic Plateau: the heuristic plateau of state $s \in S$, denoted $P_h(s)$, is the set of all states $s' \in S$ such that $h(s) = h(s')$

Definition 1.2. Connected Heuristic Plateau: the connected heuristic plateau of state $s \in S$, denoted $P_h^c(s)$, is the set of all states $s' \in S$ such that $h(s) = h(s')$ and s' is reachable from s using only states $s'' \in P_h(s)$

Definition 1.3. Heuristic Crater: the heuristic crater of a state $s \in S$, denoted $C_h(s)$, is the set of all states $s' \in S$ such that $h(s') \geq h(s)$

Definition 1.4. Connected Heuristic Crater: the connected heuristic crater of a state $s \in S$, denoted $C_h^c(s)$, is the set of all states $s' \in S$ such that $h(s') \geq h(s)$ and s' is reachable from s using only states $s'' \in C_h(s)$.

For some state $s \in S$, $P_h(s) \subset C_h(s)$ and $P_h^c(s) \subset C_h^c(s)$. A connected plateau is the *flat* region at the bottom of a crater.

2 Breadth-Based Diversification (Path Sensitivity)

One shortcoming of many type-based diversification strategies is that they don't distinguish states by breadth; only notions of depth (h, g, d) are used to bucket nodes, but there is no path sensitivity. For example, nodes in Fan's Type-Based Exploration, are bucketted by the tuple $\langle g, h \rangle$, meaning nodes in the same bucket are on the same plateau, but not necessarily on the same connected plateau. This lack of path sensitivity can lead to pathological behaviour in many exploration techniques due to the large numbers of nodes that end up getting bucketted together when bucketted globally.

The following experiments are meant to empirically assess the importance of path sensitivity in stochastic exploration techniques.

2.1 Biased Exploration with Path Sensitivity

This experiment will compare Kuroiwa's Biased Exploration, denoted Softmin-Type(h), with biased inter-plateau exploration and uniform intra-plateau exploration over the HI partitioning system, denoted HI-Softmin-Uniform(h). Both will use queue alternation with a minimum heap over h.

Softmin-Type(h) works by bucketting nodes in the same way as Fan's, but during selection it will first choose an h in a biased way (preferring low h values), and then choose uniformly from the buckets with that h-value (which means uniformly choosing a g-value from the candidate buckets). HI-Softmin-Uniform(h) will select types in a near identical way, with the only difference being that the g-value isn't used in the bucket key, but both will still select a bucket uniformly from the candidate buckets according to the selected h.

This experiment seeks to explore the idea that Kuroiwa's Biased Exploration might benefit from path sensitivity. And more generally, it seeks to investigate the idea that the pathological behaviour path sensitivity helps avoid is common enough for it to be a useful wrinkle to introduce to an exploration technique.

2.2 Type-Based Exploration with Path Sensitivity

This experiment will compare Fan's Type-Based Exploration, denoted Type-Based(h), with uniform inter-plateau exploration and uniform intra-plateau exploration over the HI partitioning system, denoted HI-Uniform^2(h). Both will use queue alternation with a minimum heap over h.

Type-Based(h) works by bucketting nodes according to the tuple $\langle h, g \rangle$ and during selection a bucket will be chosen uniformly at random, and then choose a node uniformly from that bucket. HI-Uniform^2(h) will partition nodes by HI and during a selection a random bucket will be chosen followed by a random node from that bucket.

Like above, this experiment seeks to explore the idea that Type-Based exploration might benefit from path sensitivity. And more generally, it seeks to explore the idea that path sensitivity is a useful approach in exploration.

3 Inter- and Intra- Plateau Exploration

Asai *et al.* noted the orthogonal and complementary nature of intra- vs. inter-exploration and demonstrated their orthogonality empirically.

In an attempt confirm their results and investigate the relative importance of inter- vs. intra-exploration, the HI partitioning system will be used to investigate the relative importance of the two approaches.

3.1 Inter vs. Intra Plateau Exploration

To start, I want to demonstrate that both inter- and intra-plateau exploration are independently beneficial to GBFS. To do this, I will compare *pure inter-exploration*, *pure intra-exploration*, and GBFS. *Pure inter-exploration* is HI-Uniform-Greedy(h), while *pure intra-exploration* works a bit different: at each exploration step, the type of the minimum h-valued state is selected, and then a node from that type is chosen uniformly at random; this will be denoted HI-Greedy-Uniform(h). Both will use alternation with a minimum heap.

This experiment will hopefully demonstrate that GBFS benefits from diversification both for inter-plateau exploration and intra-exploration. I do expect *pure intra-exploration* to perform worse due to the possibility for pathological behaviour in doing greedy type select (see section 4).

3.2 Combined Inter- and Intra- Plateau Exploration

This experiment will hopefully confirm Asai's result that inter- and intra- are orthogonal and complementary. HI-Uniform-Uniform(h) with alternation should outperform GBFS, HI-Greedy-Uniform(h) and HI-Uniform-Greedy(h).

3.3 Intra-Plateau Exploration

In this experiment I want to start understanding the relative importance of intra-plateau exploration along with how much is too much. To do this, 3 versions of HI-Uniform-Epsilon(h) will be run. In all cases the inter-plateau step will choose a type uniformly at random, for the second step, with probability ϵ choose a random node from the type, otherwise choose the minimum node from the type. This will be tested with $\epsilon = 0.2, 0.5, 1.0$ giving HI-Uniform-0.2(h), HI-Uniform-0.5(h), and HI-Uniform-1.0(h) respectively.

4 Pathological Behaviour of Greedy Type Selection

Greedily choosing a type should sometimes result in pathological behaviour. If the selected type does not lead to a goal node than its continued selection will result in exhausting all nodes reachable from that type until it and all generated types are empty allowing the search to finally select a type that can actually lead to a goal. To test this idea, I will compare 3 instances of HI-Epsilon-Uniform(h), setting $\epsilon = 0.0, 0.01, 0.5$.

The expected behaviour of here is that setting $\epsilon = 0.0$ will result in many searches timing out or running out of resources as they try to exhaust unproductive paths. Setting 0.01, should partially

mitigate this, but at 1% exploration I still expect many timeouts. Setting $\epsilon = 0.5$ should perform the best.