Introduction to Hypergraphs

Otherwise known as “What I did last summer...”

Review Basic Enumerable & Indexable Types

- Arrays
- Lists, Queues, Stacks
- Binary-Trees, N-way Trees
- Directed & Undirected graphs

Nomenclature of hypernets and sets

The nomenclature surounding these this complex mathematical objects is varried, the first appearances assign the name "hyper-graph" as an extention on the graph-theory at the time (1960-80) Citations are in the mail.

Hyper-*net*, multigraph, multiple-edges,multiple-node-graph, intersection-graph, Clique graph, 3-sat microstructure complements, permutation set, power set, combination set, multi-sample-knapsack problem, all solutions to all problems from:

$$O(c) \leq O(N) \leq O(N^c) \leq O(c^N) \leq O(N^N) \leq O(N^M)$$

Applications of hypergraphs

- Accuratly model the many body problem in physics.
- Model all reactions given a set of compounds.
- Model all phone calls in a telephony network.
- All cycles in a standard graph
- Roughly corresponds to writing a struct or class in programming.
- Related to group theory in mathematics.

Dense vs fully connected graph vs hypernet

- A dense graph has up to N^2 edges
- A fully connected graph has N^2 edges.
- A fully connected unrestricted hypergraph has N^∞ edges.
- A hyper-edge contains a set of nodes that can repeat.
- A hyper-graph contains the set of all nodes.
- A hyper-net contains all possible solutions to the problem.
- Finding the correct ordering of nodes & edges for the given hypergraph instance is equivelent to checking the 'set of all answers' for the correct answer for your problem.

Hypernet as a representation

- As hypernets contain an infinite number of edges; thus no 'real' instance can be created.
- A hypernet model can be built to access the infinite edges.
- A hypergraph is an instance that maps indexes to nodes and nodes to indexs.
- An odometer is an instance of an indexed set of numbers.
- A hyper-edge is an instance of an indexed set of nodes
- Odometers correspond $1:1$ with hyper-edges given a hypergraph.
- Elegant mechanicst: code reduces to a vector of numbers, vector of nodes, and functions to translate.
- Access time is constant/consistant in $\frac{size(hyper-edge)}{count(processors)}$.

Hypernet, graph, edge, and odometer template listing

```
template <class T>
class Odometer: std::vector<int>(){}

template <class T>
class HyperEdge: std::vector<T>(){}

template <class T>
class HyperGraph
{
public:

    //Index of number to index of things.
    HyperEdge<T> Lookup(Odometer<T> odometer)
    {
       HyperEdge<T> returnValue;
       returnValue.resize(odometer.size());
       for( size_t i = 0; i < odometer.size(); i++)
       {
         // this whole loops can be parallelized.
         returnValue[i] = Nodes[odometer[i]]);
       }
       return returnValue;
    }

    // Index of things to index of numbers.
    Odometer<T> Lookup(HyperEdge<T> hyperedge)
    {
       Odometer<T> returnValue;
       returnValue.resize(odometer.size());
       for( size_t i = 0; i < hyperedge.size(); i++)
       {
         // this whole loops can be parallelized.
         returnValue[i] = Lookup[hyperedge[i]]);
       }
       return returnValue;
    }

    // Evaluate a steping function from state N to N+1, and return true if there is more.
    bool EnumerateStep( bool (*func)(HyperEdge<T> item, Odometer<T> &index, HyperGraph<T> &graph),
                        Odometer<T> &current,
                        OrderedHyperGraph<T> &graph)
    {
        return (*func)(graph.GetHyperEdge(current), current,graph);
    }

private:
    // 2 N Storage time is the constant cost for infinite simulation.
    HyperEdge<T> Nodes;
    std::map<T,int> Lookup;
};
```

Directed vs. undirected vs. ordered

- Terminology for lists {push,pop,indexof} is not used for tree operations.
- Nomenclature from trees {Child, Sibling, Parent, ect} is not used in graphs.
- Directed and Undirected terminology originates from graphs
- Hypergraphs inherently rely upon indexes of numbers to access hyperedges/odometers.
- The odometer ordering determines the hyperedge ordering.
- The hyperedge ordering determines the odometer ordering.

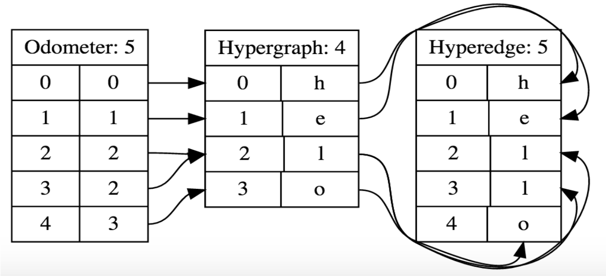
Ordered hypergraphs

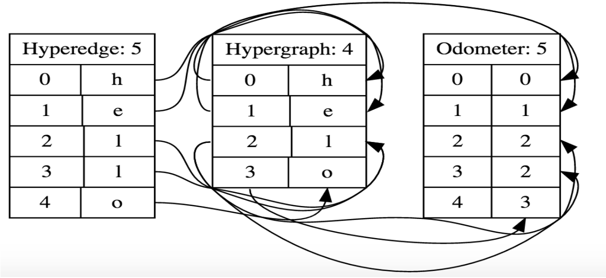
- Terminology for lists {push,pop,indexof} is not used for tree operations.
- Nomenclature from trees {child, sibling, parent, ect} is not used in graphs.
- Directed and undirected terminology originates from graphs
- Hypergraphs inherently rely upon indexes of numbers to access hyperedges/odometers.
- The odometer ordering determines the hyperedge ordering.
- The hyperedge ordering determines the odometer ordering.

Ordered hyperedges

- Permutations can be reduced to vombinations.
- Hyper-edges can be explored via adding additional nodes to them and translating back to odometer.
- Odometers can be explored via adding / subtracing numbers and translating them back to hyperedges.
- Modulo memory space turns indexing past the end into repeated data.
- Rounded memory space turn negative index into repeated data.

Converting odometers & hyperedges given a hypergraph





Explicit control of odometer determines next state to visit

- Enumerating all hyperedges whose odometer size is 2 is equivelent to a fully connected graph.
- Accessing each hyperedge is constant time in the size/count of the odometer/hyperedge $O(c)$.
- As there are an infinite number of edges, we cannot visit them all as $O(\infty * c) \rightarrow O(\infty)$
- Only guranteed to terminate functions are provided.
- Enumeration of hyperedges & odometers is now the programmers responsibility.
- Similar to std::begin() and std::end() for loop control...
- Except the function only returns true/false as to if execution has completed. And the programmer needs to write it.

Explicit odometer control

- Control over the odometer / number stack allows the transition from *ANY* hyperedge to *ANY* hyperedge in one step.
- Depth first, Breadth first search are trivial counter incrementers.
- Enumerating combinations, permutations, and multi-select-variants is a trivial task.
- Controling of depth and breadth / bound and branch is possible via external heuristics.
- Exploring the depth and breadth from a given set of nodes is a simple as converting to hyperedge, then to odometer and exploring numerically instead of quantitatively.