Definitions, Propositions, Relationships, Shorthand Syntax, and Plans of Approach

**Definitions:**

"Equivalence on Paths 1": The sorted matrix that represents the values of paths (the Paths Matrix or Paths Object), of two input graphs, is identical up to P = V, where V is the number of vertices in each one of the graphs.

"Equivalence on Paths 2": Same as Equivalence on Paths, but with P = 2V.

"Agreement on Paths": Same as Equivalence on Paths, but with P defined in terms of any polynomial function of V.

"Probable Equality on Paths": Same as Agreement on Paths, but with P is any calculable value.

"Co-Paths Graphs": The two input graphs are have probable equality on paths, but are not isomorphic to one another.

"Subgraphs": If not made more clear, this refers to the set of V graphs generated by copying G V times, and removing one vertex in each one of the copies.

**Shorthand:**

Iso(G, H) = The graphs G and H are isomorphic to one another.

PEquiv1(G, H) = The graphs G and H are Equivalent on Paths 1.

PEquiv2(G, H) = The graphs G and H are Equivalent on Paths 2.

PAgree(G, H) = The graphs G and H Agree on Paths.

PEquals(G, H) = The graphs G and H have Probable Equality on Paths.

CoPaths(G, H) = The graphs G and H are Co-Paths graphs.

SubGs(G) = The set of subgraphs of the graph G.

Connections(a, P) = The set of connections between a vertex a and a disjoint paths object P.

GI = The abstract Graph Isomorphism Problem, or its computational complexity class

P = The Polynomial time complexity class.

PathsGroups(G) = Set of quazi-equivalence classes created by paths with high discrimination.

AutoGroups(G) = Set of automorphism classes of a graph G.

Calc(T) = The complexity class of the calculating the abstract task T.

Embedded(G) = The embedded subgraph of G generated by linking its true equivalence classes.

**Propositions:**

PEquiv1(G, H) ⇒ PEquiv2(G, H)

If two graphs are equivalent on paths values P=1-V, then they are equivalent for values P=1-2V.

PEquiv2(G, H) ⇒ PAgree(G, H)

If two graphs are equivalent on paths values P=1-2V, then they are equivalent for P being any polynomial function of V.

PAgree(G, H) ⇒ PEquals(G, H)

If two graphs are equivalent on paths for any polynomial function, then they are equivalent for all observable values of P.

PEquals(G, H) ⇒ Iso(G, H)

If the paths of two graphs are identical for all observable values, then the two graphs are isomorphic.

¬∃ CoPaths(G, H)

There do not exist graphs which are probabilistically equal in Paths, but are non-isomorphic.

GI ⊂ P

The Graph Isomorphism subspace in algorithmic complexity is within the broader polynomial class.

PEquals(G, H) ⇒ (Paths(SubGs(G)) = Paths(SubGs(H)))

If the paths generated by two graphs G and H are fundamentally the same, then all of the paths of their subgraphs are the same, with a one to one correspondence enforced by the equal cardinality of the sets.

(Paths(SubGs(G)) = Paths(SubGs(H))) ⇒ PEquals(G, H)

If all of the subgraphs' paths of two graphs, G and H, are the same, then the paths of the two graphs are the same.

PEquals(G, H) ⇒ ∃ g ∈ SubGs(G), h ∈ SubGs(H) | Paths(g) = Paths(h)

If two graphs share a paths object, then there exist a pair vertices (one from each graph), such that when they are removed, the two resulting subgraphs also share a paths value (either through isomorphism or copaths).

Connections(a, Paths(A)) ∧ Paths(A) ⇒ Paths (A ∪ a)

Connections(a, Paths(A)) ∧ Paths (A ∪ a) ⇒ Paths (A)

Paths (A ∪ a) ∧ Paths(A) ⇒ Connections(a, Paths(A))

(Above three with a limited to k connections)

PathsGroups(G) = AutoGroups(G)

The groups generated through paths object comparison are fundamentally the same as the true automorphism groups of a graph.

Calc( Iso(G, H) | AutoGroups(G) ∧ AutoGroups(H) ∧ G ∧ H) ∈ P

We can create an algorithm that operates in polynomial time, which can transform automorphism groups of two graphs into an algorithm which deterministically calculates the presence of an isomorphism between the two graphs, and reports one such isomorphism.

Calc(AutoGroups) ∈ P ⇒ GI ∈ P

If we can calculate automorphism groups of a graph in polynomial time, then Graph Isomorphism is in polynomial time.

∃ G v=V, e=E, ∀ pv ∈ Paths(G), pv = k ⇒ ∃=1 such graph

If all of the vertices of a graph with a set number of edges and vertices are in the same paths group, then there does not exist another graph that satisfies the same constraints: i.e. these constraints demand that there is either 1 or 0 graphs of this type.

∀ G, H, CoPaths(G, H) ⇒ Embedded(G) != Embedded(H)

The embedded subgraphs (formed by analysis of the structure of the Paths Groups) of all co-paths graph pairs must be fundamentally distinct

Paths(SubGs(G))=Paths(SubGs(H)) ∧ Paths(G)=Paths(H) ⇒ Iso(G, H)

Paths performs worse on medium connected graphs.

Paths performs worse on larger numbers of vertices.

Paths performs worse on even values of V.

Paths performs worse on even values of E.

Paths performs worse means that we are more likely (by probability) to find co-paths, non-isomorphic graphs within the given subset.

Paths(G) ⇒ Polygon(G)

If we are given the paths of a graph, for some fixed power, we can find the corresponding values of polygons, through manipulation of the paths values.

∃ ρ | Paths(G, P) 1≤P≤ρ ⇒ Paths(G, P) 1≤P≤∞

There exists some maximal value of power, such that if we have all the paths values beneath that value, we can use them to construct the paths values above them with determinism.

**Symbols:**

∩→←⇒⇐⇔∞≤≥∪⊃⊇⊄⊂⊆∈∉∧¬∅∨∃∀