Graph algorithms are represented as the iterative execution of the same piece of code on every vertex.

Representative Software
Pregel: A synchronous vertex-centric graph processing framework.

Pregel = BSP + Vertex-Centric Approach

BSP superstep
- high scalability
- universality

Bulk-Synchronous Parallel (BSP) model
- message-passing is hard
- vertices have to maintain complicated states

Pregel = BSP + Vertex-Centric Approach
- remote access capabilities
- Chain access (new feature)
- Remote writes (enhanced feature)
- Neighborhood aggregation

let t = [ P[e.ref] | e <- Nbr[u] ]

Extending Green-Marl and Fregel’s declarative programming models

Strength of Pregel
- Pattern matching
- Pattern writing
- Pattern accessing
- Pattern reading
- Pattern writing

Example: pointer-jumping

Pregel = iteration construct
+ vertex-centric model
+ remote access

Palgol: a domain-specific language with remote access capabilities

Key technique: compiling general chain access

Logic system for handling P[P[P[u]]]:
1. \( \forall u.K_v\ u \) every vertex \( u \) knows its own identifier \( u \)
2. \( \forall u.K_vP[u] \) every vertex \( u \) knows its local field \( P[u] \)
3. \( (\forall u.K_w(e(u)) \land (\forall u.K_{w[u]}v(u))) \Rightarrow \forall u.K_{v[u]}e(u) \)
   we can make every \( v(u) \) know \( e(u) \) by letting every \( w(u) \) send \( e(u) \) to \( v(u) \)

Step 1: \( u \) knows \( u \)
- \( u \) knows \( P[u] \)
- message passing
- logical inference

Step 2:
- \( P[u] \) knows \( u \)
- \( P[u] \) knows \( P[P[u]] \)

Step 3:
- \( u \) knows \( P[P[u]] \)
- \( P[P[u]] \) knows \( u \)
- \( P[P[u]] \) knows \( P^4[u] \)

Step 4:
- \( u \) knows \( P^4[u] \)
- the derivation of chain access \( P[P[P[u]]] \)

A backtracking algorithm with memoization for finding a solution with minimum number of steps

Application: the pointer-jumping technique
(a useful graph transformation used in S-V, MSF and so on)

Given a forest where each vertex initially knows its parent, how to let every vertex point to its root?

phase 1

phase 2

phase 3

all vertices constantly find their “grandparents”

Performance Evaluation
We compare Palgol with hand-coded Pregel programs on real-world graphs
- Minimum spanning forest (MSF)
- Strongly connected component (SCC)
- S-V connected component algorithm (S-V)

<table>
<thead>
<tr>
<th></th>
<th>SCC</th>
<th>S-V</th>
<th>MSF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.66% – 1.58%</td>
<td>-2.53% – 6.37%</td>
<td>-4.17% – 6.42%</td>
</tr>
</tbody>
</table>