

In[104]:=

```
(* ===== *)
(* TCCT 演化与费米统计验证 (全兼容修复版) *)
(* ===== *)

ClearAll[];
SeedRandom[1234];

(* --- 1. 定义核心函数 (StepEvolution) --- *)
StepEvolution[g_Graph] := Module[{
  allEdges, activeEdges, candidates,
  e1, e2, x, y, z, w,
  newActive, inertEdges, remainingEdges,
  pairFound, shuffled, nActive
},

  allEdges = EdgeList[g];
  activeEdges = Cases[allEdges, _UndirectedEdge];
  nActive = Length[activeEdges];

  If[nActive < 2, Return[g];

  shuffled = RandomSample[activeEdges];
  pairFound = False;

  (* 寻找反应对 *)
  Do[
    e1 = shuffled[[i]];
    Do[
      e2 = shuffled[[j]];
      (* 严格判定：共享恰好1个顶点 *)
      If[Length[Intersection[List @@ e1, List @@ e2]] == 1,
        pairFound = True;
        Break[];
      ];
    , {j, i + 1, nActive}];
  If[pairFound, Break[]];
  , {i, 1, nActive - 1}];

  If[Not[pairFound], Return[g];

  (* 确定坐标 *)
```

```

y = Intersection[List @@ e1, List @@ e2][[1]];
x = Complement[List @@ e1, {y}][[1]];
z = Complement[List @@ e2, {y}][[1]];
w = Max[VertexList[g]] + 1;

(* 生成新边 *)
newActive = {UndirectedEdge[x, z], UndirectedEdge[x, w], UndirectedEdge[w, z]};
inertEdges = {DirectedEdge[x, y], DirectedEdge[y, z]};
remainingEdges = Complement[allEdges, {e1, e2}];

(* 构造新图 (使用 Union 去重) *)
Graph[
  Join[VertexList[g], {w}],
  Union[remainingEdges, newActive, inertEdges],
  GraphLayout ->
]
];

(* --- 2. 定义带探针的演化函数 (Probe) --- *)
StepEvolutionWithProbe[g_Graph] := Module[{
  allEdges, activeEdges, shuffled, pairFound,
  e1, e2, x, y, z, w,
  k, accepted,
  newActive, inertEdges, remainingEdges, finalEdges,
  oldEdgeCount, newEdgeCount
},

  allEdges = EdgeList[g];
  activeEdges = Cases[allEdges, _UndirectedEdge];

  (* 状态检查 *)
  If[Length[activeEdges] < 2, Return[{g, -1, False}]];

  shuffled = RandomSample[activeEdges];
  pairFound = False;

  Do[
    e1 = shuffled[[i]];
    Do[
      e2 = shuffled[[j]];
      If[Length[Intersection[List @@ e1, List @@ e2]] == 1,

```

```

    pairFound = True;
    Break[];
  ];
  , {j, i + 1, Length[shuffled]}}];
  If[pairFound, Break[]];
  , {i, 1, Length[shuffled] - 1}}];

If[Not[pairFound], Return[{g, -1, False}]];

(* 坐标与统计 *)
y = Intersection[List @@ e1, List @@ e2][[1]];
x = Complement[List @@ e1, {y}][[1]];
z = Complement[List @@ e2, {y}][[1]];
w = Max[VertexList[g]] + 1;

(* 探针：统计反应前 x-z 之间的边数（无论死活）*)
k = Length[Select[allEdges, Sort[List @@ #] == Sort[{x, z}] &]];

newActive = {UndirectedEdge[x, z], UndirectedEdge[x, w], UndirectedEdge[w, z]};
inertEdges = {DirectedEdge[x, y], DirectedEdge[y, z]};
remainingEdges = Complement[allEdges, {e1, e2}];

(* 执行 Union *)
finalEdges = Union[remainingEdges, newActive, inertEdges];

(* 判定 accepted：看总边数变化量 *)
(* 理论：-2(消耗) + 2(冻结) + 2(必成新边xw/wz) + 1(可能成的xz) = +3 *)
(* 实际：如果 diff == 3 则 xz 成功；如果 diff == 2 则 xz 被吞 *)
diff = Length[finalEdges] - Length[allEdges];
accepted = (diff == 3);

{Graph[Join[VertexList[g], {w}], finalEdges], k, accepted}
];

(* --- 3. 运行主程序 --- *)

(* 初始化统计数据（使用普通列表，兼容旧版）*)
(* stats 结构：{k=0的次数, k=0成功数, k=1次数, k=1成功数, k=2次数, k=2成功数} *)
statsAttempts = <| 0 → 0, 1 → 0, 2 → 0, 3 → 0 |>;
statsSuccess = <| 0 → 0, 1 → 0, 2 → 0, 3 → 0 |>;

```

```

(* 初始化宇宙 *)
universe =
  Graph[{1, 2, 3}, {UndirectedEdge[1, 2], UndirectedEdge[2, 3], UndirectedEdge[3, 1]};

(* 预热：先跑 500 步让宇宙长起来 *)
Print[];
Do[universe = StepEvolution[universe], {500}];
Print["finished: ", VertexCount[universe]];

(* 开始验证 *)
validationSteps = 3000;
Print[Style["Ver_begining (2000 steps)...", Blue, Bold]];

Monitor[
  Do[
    result = StepEvolutionWithProbe[universe];

    (* 解包结果 *)
    newG = result[[1]];
    kVal = result[[2]];
    isAccepted = result[[3]];

    (* 记录有效数据 *)
    If[kVal != -1,
      If[KeyExistsQ[statsAttempts, kVal],
        statsAttempts[kVal] = statsAttempts[kVal] + 1;
        If[isAccepted, statsSuccess[kVal] = statsSuccess[kVal] + 1];
      ];
      universe = newG;
    ];

    , {step, 1, validationSteps}],

  Row[{ProgressIndicator[step, {0, validationSteps}],
    " steps: ", step, " | current k=", kVal}]
];

(* --- 4. 绘图与报告 --- *)
Print[];
probs = {};
labels = {};

```

```
kValues = {0, 1, 2};
```

```
Do[
  att = statsAttempts[k];
  succ = statsSuccess[k];
  prob = If[att > 0, succ/att, 0.0];
  AppendTo[probs, prob];
  label = ToString[succ] <> "/" <> ToString[att];

  Print["k=", k, " | attampt: ", att,
    " | success: ", succ, " | P: ", NumberForm[prob*100, {4, 2}], "%"];
, {k, 0, 2}];

BarChart[probs,
  ChartLabels → Placed[kValues, Below],
  LabelingFunction → (Placed[Row[{NumberForm[#, 100, {3, 1}], "%"}], Above] &),
  ChartStyle → {Green, Orange, Red},
  PlotLabel → "Acceptance Probability vs. Link Density (Fermi Check)",
  AxesLabel → {"Existing Links (k)", "Probability P(Add)"},
  PlotRange → {0, 1.1},
  ImageSize → 400
]
```

```
start (500 steps)...
```

```
finished: 503
```

```
Ver_begining (2000 steps)...
```

 **Cloud:** This computation has exceeded the time limit for your plan.

Out[116]=

```
$Aborted
```

```
=== Result/report ===
```

k=0 attampt: 1061 success: 893 P:	$\frac{89300}{1061}\%$
k=1 attampt: 1708 success: 916 P:	$\frac{22900}{427}\%$
k=2 attampt: 167 success: 62 P:	$\frac{6200}{167}\%$

Out[122]=

