Índice de Seguridad: un indicador de dependencia comercial directa e indirecta

Equipo Índice de Seguridad
contacto@indice-seguridad.org

20 de octubre de 2025

Resumen

1. Introducción y motivación

- Proponemos un indicador *trazable* de dependencia total (DT) desagregado por país y por industria.
- Distinguimos DD (bilateral) de DI (rutas con intermediarios) y caracterizamos "*hidden dependence*" cuando $\mathrm{DT}/\mathrm{DD} \gg 1$.
- Ofrecemos salidas operativas (rutas críticas, intermediarios) útiles para vigilancia de riesgo.

2. Relación con la literatura

3. Datos

- Universo de países: *[completar]*; industrias: *[completar]*.
- Matrices $X^{(\ell)}$ por industria ℓ: entrada $X^{(\ell)}_{uv}$ es el flujo de $u \to v$.
- Limpieza: corte relativo por columna (importador) y eliminación de columnas/filas nulas tras el corte.

4. Metodología del indicador

4.1. Dependencia directa

Para un par país destino i y origen j, definimos la dependencia directa como

$$\mathrm{DD}_{ij} \;=\; \frac{X_{ji}}{\sum_k X_{ki}}, \tag{1}$$

donde el denominador suma todas las importaciones de i.

4.2. Matriz de transición por columnas

Definimos la matriz de transición T con normalización por columnas:

$$T_{uv} \;=\; \frac{X_{uv}}{\sum_a X_{av}}, \qquad \text{con la convención } \sum_a X_{av} = 0 \Rightarrow T_{uv} = 0. \tag{2}$$

4.3. Dependencia indirecta (rutas)

Sea $\mathcal{P}_{j\rightsquigarrow i}$ el conjunto de rutas simples (sin repetir nodos) desde j hasta i con longitudes $2, \ldots, L_{\text{máx}}$. La contribución de una ruta $p = (j \to \cdots \to i)$ es el producto de transiciones:

$$\text{contrib}(p) \;=\; \prod_{(u\to v)\in p} T_{uv}. \tag{3}$$

La dependencia indirecta de i respecto a j es

$$\text{DI}_{ij} \;=\; \sum_{p\in\mathcal{P}_{j\rightsquigarrow i}} \text{contrib}(p). \tag{4}$$

4.4. Dependencia total y "hidden dependence"

$$\text{DT}_{ij} \;=\; \text{DD}_{ij} + \text{DI}_{ij}. \tag{5}$$

Diremos que existe *hidden dependence* cuando $\text{DT}_{ij}/\text{DD}_{ij} \gg 1$ para un umbral a definir (p. ej., > 2).

4.5. Intermediarios críticos

Definimos una puntuación para un país m como intermediario en rutas hacia i que parten de j combinando (i) la *frecuencia* con que m aparece en rutas de mayor contribución y (ii) la *fuerza* acumulada de dichas rutas. La métrica compuesta puede escribirse como

$$\text{IC}_{m;ij} \;=\; \alpha \cdot \text{freq}_{m;ij} \;+\; (1-\alpha)\cdot \text{strength}_{m;ij}, \quad \alpha \in [0,1]. \tag{6}$$

5. Implementación computacional (reproducibilidad)

- Precalcular T por industria; filtrar aristas con $T_{uv} < \epsilon_{\text{edge}}$.
- Para cada par (j, i), explorar rutas con DFS con poda ($\text{L}_{\text{máx}}, \epsilon_{\text{contrib}}$, `top_m`).$Paralelizarporpares(j, i)$. Almacenar: DD, DI, DT, rutas críticas y métricas de intermediación.

6. Resultados

- Distribución de DD, DI, DT por país destino e industria.
- Top 10 dependencias totales por industria y sus rutas críticas asociadas.
- Ranking de intermediarios críticos.

7. Robustez y sensibilidad

8. Discusión

9. Conclusiones

Apéndices

A. Tabla de notación

B. Pseudocódigo (DFS con poda)

```
Input: T (matriz de transición), i (destino), j (origen),
```

Símbolo	Descripción
X_{uv}	Flujo de comercio desde u (origen) hacia v (destino)
T_{uv}	Proporción del suministro de v que proviene de u (normalización por columnas)
DD_{ij}	Dependencia directa de i respecto a j
DI_{ij}	Dependencia indirecta (suma de rutas)
DT_{ij}	Dependencia total
$L_{\text{máx}}$	Longitud máxima de las rutas consideradas
ϵ_{edge}	Umbral mínimo para conservar una arista en T
$\epsilon_{\text{contrib}}$	Umbral mínimo de contribución de una ruta durante la exploración
`top_m`	Máximo de vecinos por nodo en la exploración

```
       Lmax, eps_edge, eps_contrib, top_m

function IndirectDependence(i, j):
    total = 0
    stack = [(j, 0, 1.0)]  # (nodo actual, longitud, prod acumulado)
    while stack not empty:
        (u, L, prod) = stack.pop()
        if L >= 1 and u == i:
            total += prod
        if L == Lmax: continue
        # Seleccionar vecinos relevantes (v) con T[u,v] >= eps_edge
        for v in top_neighbors(u, top_m):
            w = T[u,v]
            if w < eps_edge: continue
            prod2 = prod * w
            if prod2 < eps_contrib: continue
            if v not in ruta_actual:  # evitar ciclos
                push (v, L+1, prod2)
    return total
```