NPM Ekosisteminde Top N Paket İçin Yönlü Karmaşık Ağ Analizi: Merkeziyet, Risk Skoru ve Robustluk

Yusuf Talha ARABACI

21 Ekim 2025

Özet

Bu çalışma, NPM ekosistemindeki popüler Top N paketi, bağımlılık ilişkilerine göre yönlü bir karmaşık ağ (Dependent → Dependency) olarak modellemekte ve yapısal riskleri merkeziyet metrikleriyle incelemektedir. Veri, her çalıştırmada API'lerden (öncelikle ecosyste.ms; yedek olarak npm registry ve npms.io) çekilmektedir. Ağ, NetworkX ile kurulmakta; in-degree, out-degree ve betweenness merkeziyet metrikleri hesaplanmaktadır. Büyük grafiklerde betweenness hesaplaması örnekleme (k) ile hızlandırılmaktadır. Çalışma ayrıca bileşik bir risk skoru (normalize edilmiş in/out/between ağırlıklı toplamı) ve risk tabanlı robustluk analizi (kritik düğümlerin kaldırılması) önermektedir. Üretilen tüm çıktılar (CSV/MD/JSON ve PNG/SVG görseller) results/ dizininde saklanır.

1 Giriş

Yazılım tedarik zinciri saldırılarında (SSCA), tek bir bağımlılığın ele geçirilmesi geniş çapta zincirleme etkilere yol açabilir. NPM ekosistemi, yoğun bağımlılık ilişkilerine sahip olup, paketlerin yapısal konumuna göre sistemik risk taşıyabilmektedir. Bu çalışma, Top N paket üzerinden inşa edilen yönlü bağımlılık ağı ile aşağıdaki sorulara odaklanır:

- Hangi düğümler (paketler) yapısal olarak kritik (yüksek in-degree, yüksek betweenness)?
- Hangi düğümler geniş bağımlılık yüzeyine sahip (yüksek out-degree)?
- Merkeziyetlere dayalı bileşik bir risk skoru ile risk liderleri nasıl sıralanır?
- Kritik düğümler kaldırıldığında ağın bağlanırlığı nasıl değişir (robustluk)?

2 Veri ve Yöntem

Top N paket listesi, her çalıştırmada API'lerden çekilir. Öncelik ecosyste.ms üzerinde indirmeye göre sıralı paket adlarındadır; başarısız durumlarda npm registry araması (popularity) ve npms.io (popülerlik skoru) yedek olarak kullanılır. Bağımlılıklar, npm registry'de paketlerin en güncel sürümlerinin `dependencies` alanından okunur; isteğe bağlı olarak `peerDependencies` de dahil edilebilir. Yönlü ağ, Dependent → Dependency yönüyle kurulur. Büyük graflarda betweenness merkeziyeti örneklemeli (k) hesaplanır.

3 Ağ Modeli ve Metrikler

Model, NetworkX ile kurulmuş `DiGraph` yapısıdır. Temel metrikler:

- **In-Degree:** Düğüme gelen kenar sayısı (bu pakete dayanan paket sayısı). Ele geçirilirse etki alanını gösterir.

- **Out-Degree:** Düğümün dış bağımlılık sayısı. Bağımlılık zinciri uzunluğu/karmaşıklığına işaret eder.
- **Betweenness:** En kısa yollardaki aracılık. Köprü rolünü ve tek hata noktası potansiyelini gösterir.

4 Bulgular

Bu bölümde, results/ dizinindeki çıktıları kullanarak görsel ve tablolu özetler sunulmaktadır.

4.1 Ağ Görselleştirmeleri

network_full_topN.png bulunamadı

Şekil 1: Top N + bağımlılıkların oluşturduğu yönlü ağ (düğüm boyutu: in-degree, renk: Top N turuncu / diğerleri mavi).

network_topN_only.png bulunamadı

Şekil 2: Sadece Top N düğümlerin indüklenmiş alt-ağı.

4.2 Derece Dağılımları ve Korelasyonlar

degree_histograms.png bulunamadı

Şekil 3: In-Degree ve Out-Degree histogramları (log ölçek).

4.3 Liderler (İlk 10)

4.4 Ek Görseller

4.5 Risk Skoru ve Robustluk

Edge Betweenness İlk 10. Aşağıdaki tablo, en yüksek edge betweenness değerine sahip 10 kenarı göstermektedir.

4.6 Ek Tablolar

Top 20 In-Degree (Tüm Düğümler).

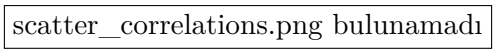
Top 20 Risk Skoru.

5 Risk Skoru Tanımı

Normalize edilmiş metrikler (min-max) üzerinden ağırlıklandırılmış bir risk skoru tanımlanmıştır:

$$\text{risk}(n) = w_{in}\,\tilde{d}_{in}(n) + w_{out}\,\tilde{d}_{out}(n) + w_b\,\tilde{b}(n),$$

burada $\tilde{d}_{in}, \tilde{d}_{out}, \tilde{b}$ sırasıyla in-degree, out-degree ve betweenness'in normalize edilmiş değerleridir. Ağırlıklar (ör. $w_{in} = 0.5, w_{out} = 0.2, w_b = 0.3$) senaryoya göre ayarlanabilir.



Şekil 4: Korelasyonlar: In-Degree vs Betweenness (solda), In-Degree vs Out-Degree (sağda).

Şekil 5: İlk 10 lider (In/Out-Degree ve Betweenness).

6 Robustluk Analizi

Risk skoruna göre en kritik $k \in \{1, 3, 5\}$ düğüm kaldırıldığında zayıf bağlanırlık bileşen sayısı, en büyük bileşen boyutu ve (mümkünse) çap raporlanır. Ayrıntılar results/ dizinindeki `robustness_risk.json` dosyasındadır.

7 Sınırlamalar ve Gelecek Çalışmalar

Sınırlamalar: (i) Varsayılan olarak yalnız `dependencies` kullanılır; `peerDependencies` isteğe bağlıdır. (ii) Betweenness hesaplaması büyük graflarda örneklemelidir; kesin değerlerin yakınsaması ağı büyüklüğüne ve k seçimine bağlıdır. (iii) Global dependent sayıları doğrudan dahil edilmemiştir.

Gelecek Çalışmalar: (i) Global dependent metriklerinin entegrasyonu, (ii) ağırlıkların veri odaklı (öğrenilmiş) belirlenmesi, (iii) çok katmanlı ağ modelleme (örn. dev/peer/optional bağımlılık katmanları), (iv) GEXF/GraphML çıktılarıyla etkileşimli görselleştirme.

8 Sonuç

Bu çalışma, NPM ekosistemindeki popüler paketler için yönlü bağımlılık ağını kullanarak yapısal riskleri nicel olarak analiz etmektedir. Merkeziyet metrikleri, bileşik risk skoru ve robustluk analizleri, kritik düğümlerin ve köprülerin pratikte nasıl belirleneceğine dair net bir çerçeve sunmaktadır.

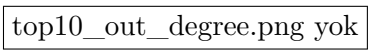
Çoğaltılabilirlik. Tüm kod ve çıktı üretimi rapoyla birlikte depo içindedir. `analysis.ipynb` defteri çalıştırılarak results/ dizini yeniden üretilebilir ve bu makale LaTeX dosyası ile raporlanabilir.

Kaynaklar

[1] Newman, M. (2010). *Networks: An Introduction*. Oxford University Press.

[2] Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*.

[3] Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.



Şekil 6: İlk 10 In-Degree (sol) ve Out-Degree (sağ).

Şekil 7: İlk 10 Betweenness.

Şekil 8: Bileşik risk skoruna göre ilk 20 paket.

Tablo 1: Top 20 In-Degree (Toplam Düğümler)

Paket	In-Degree	Out-Degree	Betweenness	TopN?
tslib	62	0	0.000000	True
@babel/helper-plugin-utils	58	0	0.000000	True
@smithy/types	48	1	0.000001	True
call-bound	38	2	0.000183	True
es-errors	28	0	0.000000	True
call-bind	24	4	0.000121	True
@jest/types	23	7	0.000207	True
@types/node	23	1	0.000034	True
chalk	23	0	0.000000	True
debug	21	1	0.000051	True
@aws-sdk/types	20	2	0.000002	True
jest-util	20	6	0.000099	True
@babel/types	18	2	0.000079	True
define-properties	18	3	0.000043	True
graceful-fs	18	0	0.000000	True
get-intrinsic	17	10	0.000328	True
es-object-atoms	15	1	0.000006	True
gopd	15	0	0.000000	True
@smithy/protocol-http	14	2	0.000000	True
semver	14	0	0.000000	True

Tablo 2: Top 20 Risk Skoru

Paket	Risk	In-Degree	Out-Degree	Betweenness	TopN?
es-abstract	0.534931	10	54	0.000785	True
tslib	0.500000	62	0	0.000000	True
@babel/helper-plugin-utils	0.467742	58	0	0.000000	True
@smithy/types	0.390249	48	1	0.000001	True
call-bound	0.382129	38	2	0.000183	True
jest-snapshot	0.303511	5	21	0.000532	True
get-intrinsic	0.290993	17	10	0.000328	True
@jest/types	0.284735	23	7	0.000207	True
@jest/transform	0.251456	6	15	0.000419	True
call-bind	0.251171	24	4	0.000121	True
@babel/traverse	0.231984	13	7	0.000280	True
es-errors	0.225806	28	0	0.000000	True
jest-util	0.216164	20	6	0.000099	True
@types/node	0.201331	23	1	0.000034	True
@babel/preset-env	0.200000	0	70	0.000000	True
@babel/core	0.199075	4	15	0.000324	True
debug	0.191845	21	1	0.000051	True
@jest/core	0.187008	2	28	0.000238	True
chalk	0.185484	23	0	0.000000	True
@babel/types	0.181088	18	2	0.000079	True