SUPPLEMENT

**Drivergene.net: A Cytoscape app for the identification of driver nodes of large-scale complex networks and case studies in discovery of drug target genes.**

**Duc-Tinh Pham1,2*,*** **Tien-Dzung Tran1,3\***

1Complex Systems and Bioinformatics Lab, Hanoi University of Industry, Hanoi, Vietnam.

2Gruduate University of Science and Technology, Vietnam Academy of Science and Technology, Hanoi, Vietnam.

3Faculty of Information and Communication Technology, Hanoi University of Industry, Hanoi, Vietnam.

\*Corresponding: TD Tran ([trantd@](mailto:trantd.vn@)haui.edu.vn)

Supplementary Information

# Drivergene.net app

Drivergene.net is integrated with Cytoscape through Application Programming Interface (API). This software, developed in a Java programming environment, is compiled, and packaged separately as a plug-in for Cytoscape, a popular network analysis software. It can be run on any system with a programming language compiler installed for Java. Details of the features, installation steps, and instructions for use of Drivergene.net software are presented in this document.

|  |
| --- |
| Graphical user interface, text, application  Description automatically generated |
| Fig. 1. Screenshot of Drivergene.net. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **Algorithm 1.** Parallelly finding driver nodes of a complex network *G*(*V*, *E*). This algorithm is applied in the identification of anticancer drug target genes from biomolecular networks. | | | |
| 1 | **function** *[Xt]* *InsideCompetition(G(V,E), Leaders ⊂ V, AgainstLeaders ⊂ V)* | 27 | *Support* ←**new** *Dictionary<node,value>* |
| 2 | *maxIterations*←*100000; Epsilon ← 4.94e-324* | 28 | *β* ←**new**Node |
| 3 | *ε* ←*1/(Max*(total weights of out-links of *v*, ∀*v*∈*V) – Epsilon* | 29 | *NormalAgents ← V \{β,*α*}* |
| 4 | *Xt*←**new** *Dictionary<node,value>; Xt+1* ←**new** *Dictionary<node,value>* | 30 | **for** (*γ* **in** *NormalAgents* ) **do** |
| 5 | **for** (*Node* **in** *V* ) **do** | 31 | *e←* **new** *Edge(β, γ); E= E ∪ {e}; ← InsideCompetition(G(V,E), {*α*},{β});* |
| 6 | *Xt[Node]* ←*0* | 32 | *Support[γ] ← X[γ]; E= E \ {e}* |
| 7 | **for** (*Leader* **in** *Leaders*) **do** | 33 | **return** *Support* //Support of nodes to α when connecting to *β* |
| 8 | *Xt[Leader]* ←*1; Xt+1[Leader]* ←*1* | 34 | **end** |
| 9 | **for** (*AgainstLeader* **in** *AgainstLeaders*) **do** | 35 | **procedure** *ToS(G(V,E),* α *∈ V,* **out** *result)* |
| 10 | *Xt[AgainstLeader] ← -1; Xt+1[AgainstLeader]←* -1 | 36 | *Support←* **new** *Dictionary<node,value>* |
| 11 | *Error ← 0; t ← 0* | 37 | *Support← OutsideCompetition (G(V,E),* α*)* |
| 12 | **do** | 38 | *TotalSupport ← 0* |
| 13 | *Error ← 0* | 39 | **for** (*γ* **in** *V*\{α}) **do** |
| 14 | **for** (*u* **in** *V*) **do** | 40 | *TotalSupport ← TotalSupport + Support[γ]* |
| 15 | **if***(Leaders* contain *u* **or** *AgainstLeaders* contain *u)* **then****continue**; | 41 | *result*[α]= *TotalSupport* //Total support of nodes to α |
| 16 | *r←0;* | 42 | **end** |
| 17 | **for** (*v* **in** *Neighbors of u*) **do**; | 43 | **function**  *[driver nodes] ParFindDriverNode(G(V,E))* |
| 18 | *r ← r + weight(u,v)\*(Xt[v]- Xt[u]);* | 44 | //*G(V,E):* Global variable |
| 19 | *Xt+1[u]* ← *Xt[u]+ ε\* r;* | 45 | *//α:* Local variable |
| 20 | *Error ← Error + Math.Abs(Xt[u] - Xt+1[u]);* | 46 | *//result:* Local variable |
| 21 | *Temp* ←*Xt; Xt*← *Xt+1; Xt+1*←*Temp* | 47 | *result ←* **new** *Dictionary<node,value>* |
| 22 | *t* ← *t +1* | 48 | **for(***α* **in** *V*) **in parallel do** |
| 23 | **while** *(Error > Epsilon && t < maxIterations)* | 49 | *ToS(G(V,E),α, result)* |
| 24 | **return** *Xt*; //Output as stable states of nodes as *t*→ ∞ | 50 | wait for all works done() |
| 25 | **end** | 51 | **return** *result* |
| 26 | **function** *[Support]* *OutsideCompetition (G(V,E),* α *∈V)* | 52 | **end** |