SmartGraph API: Programmatic Knowledge Mining in Network-Pharmacology Setting

Gergely Zahoránszky-Kőhalmi[1,*], Brandon Walker[1], Nathan Miller[1], Brett Yang[1], Dhatri V. L. Penna[1], Jessica Binder[1], Timothy Sheils[1], Ke Wang[1], Jennifer King[1], Hythem Sidky[1], Sridhar Vuyyuru[1], Jeyaraman Soundarajan[1], Samuel G. Michael[1], Alexander G. Godfrey[1], Tudor I. Oprea[2,3,4,5]

[1]National Center for Advancing Translational Sciences (NCATS/NIH), 9800 Medical Center Dr., Rockville, MD 20850, USA

[2]Department of Internal Medicine, University of New Mexico School of Medicine, 1 University of New Mexico, Albuquerque, NM 87131, USA

[3]Novo Nordisk Foundation Center for Protein Research, Faculty of Health and Medical Sciences, University of Copenhagen, Blegdamsvej 3B, 2200 Copenhagen N, Denmark

[4]UNM Comprehensive Cancer Center, 1201 Camino de Salud NE, Albuquerque, NM 87102, USA

[5]Department of Rheumatology and Inflammation Research, Institute of Medicine, Sahlgrenska Academy at University of Gothenburg, Box 480, 40530 Gothenburg, Sweden

*Corresponding author:

Gergely Zahoránszky-Kőhalmi

Email: gergely.zahoranszky-kohalmi@nih.gov

ABSTRACT

The recent SmartGraph platform facilitates the execution of complex drug-discovery workflows with ease in the network-pharmacology paradigm. However, at the time of its publication, we identified the need for the development of an Application Programming Interface (API) that could promote biomedical data integration and hypothesis generation in an automated manner. This need was magnified at the time of the COVID-19 pandemic. This study addresses this hiatus. Most functionalities of the original platform were implemented in the SmartGraph API. We demonstrate that by using the API it is possible to transform the original semi-automated workflow behind the Neo4COVID19 database to a fully automated one. The availability of the SmartGraph API lends a significant improvement to the programmatic integration of network-pharmacology-oriented knowledge graphs and analytics, as well as predictive functionalities and workflows.

INTRODUCTION

The “SmartGraph network-pharmacology investigation platform” [1] was designed to execute complex drug-discovery-related workflows with ease. The frontend of the platform was developed with this mindset and gave rise to an interactive web-based graphical user interface (GUI). The GUI facilitates effective data exploration, visualization, and workflow execution, as well as the computation of bioactivity predictions via so-called “potent patterns” [1]. Due to the design considerations of SmartGraph, these functionalities of the platform are accessible to anybody without an extensive background in cheminformatics and bioinformatics.

At the time of publishing SmartGraph, in January 2020, we identified the need for a standalone SmartGraph API that provides programmatic access to most of the functionalities that are accessible via the frontend (see: “Conclusions” in *Zahoranszky-Kohalmi et al*. [2]). Little did we know how soon this need would be magnified by the onset of the COVID-19 pandemic, only weeks after the publication of SmartGraph. The pandemic shed light on the pivotal role of rapid data integration and computation-driven hypothesis generation, which were identified as bottlenecks in the translational process [2].

Indeed, our recent work on a semi-automated workflow that addresses biomedical data integration for COVID-19 research demonstrated the drawbacks of lacking an API. Namely, this hiatus of the SmartGraph platform required us to execute certain steps in the workflow manually. This approach is not scalable and is quite error-prone. Therefore, we decided to develop an

Application Programming Interface (API) for SmartGraph which provides access to most functions available for the investigators via the GUI. We demonstrate how this improvement enables the execution of the same workflow in a completely automated manner. Furthermore, we hope that this effort further emphasizes the need for the development of APIs, workflows, and standards as best practices, which will be critical for a rapid response to emerging pathogens.

Prior Art

Before this study, no API existed for the SmartGraph platform that we know of. There exists, however, other platforms with APIs that can be utilized in a network pharmacology setting. The STRING database [3] includes experimental determined and predicted protein-protein (regulatory) interactions. DrugCentral [4, 5], ChEMBL [6, 7], PubChem [8, 9] and DrugBank [10, 11] contain drug-target interaction data. The KEGG [12], Pharos [13-16], and RampDB [17, 18] databases include pathway information besides protein targets and small-molecule (drug) interaction data. While these databases are essential sources to build knowledge graphs, such as Neo4COVID19, and Hetionet [19, 20], from a knowledge mining perspective it is pivotal to have a path search functionality. The only platform that provides such functionality as well as an API is the proprietary Ingenuity Pathways Analysis (IPA) platform.

While the SmartGraph platform shares similar features with the aforementioned resources, it has a unique network analysis-driven bioactivity predictive capability on the basis of “potent patterns” [1], that no other resources are equipped with, to the best of our knowledge.

RESULTS AND DISCUSSION

SmartGraph API

The underlying architecture of the original SmartGraph platform reflects a modular view-controller-data design pattern, where the frontend (web-based graphical user interface) makes database queries via the Bolt protocol to the backend (Neo4j graph database) [21]. While the design enables a responsive graphical user interface (GUI) with efficient data visualization, it failed to support the integration of SmartGraph data and embedded drug-discovery logic into arbitrary workflows via programmatic access. Luckily, with the advent of containerization technologies and API development frameworks, we were able to address this hiatus.

We utilized the FastAPI Python-based API development framework to implement the logic of SmartGraph API endpoints [22, 23]. The advantages of this framework include the ease of implementation, support for asynchronous calls, and the automated generation of API documentation. The documentation is exposed via the de-facto industry standard SWAGGER interface [24].

The logic in API endpoints was developed with the help of graph database queries using the Cypher language [21]. The execution of queries and collection of results were handled by the official "neo4j" Python driver [25] for efficient data transfer via the Bolt protocol.

The containerization technology enables the packaging of the SmartGraph API into a Docker container. The environment for the FastAPI is provided by the Conda environment manager. This implementation simplifies the deployment process of the SmartGraph API and it facilitates scalability, as needed.

API Functionalities

The functionalities of the SmartGraph API can be categorized into four main categories (see: *Fig 1.*). Based on bioactivity data collected from ChEMBL, endpoints falling under the "bioactivities" category, can find the "active" compounds of a protein target or find targets that a compound is "active" on. The names of those endpoints are "/bioactivity_target" and "/bioactivity_compound", respectively. The definition of "active" reflects a relationship between a target and a compound, and it can be considered as an aggregated bioactivity value more potent, than a predefined cutoff. The choice of the cutoff is left to the investigator, and it is controlled via an endpoint argument. Most endpoints were designed to support batch queries. That is, it is possible to submit a list of targets or compounds for which the "active" bioactivity relationships are sought after.

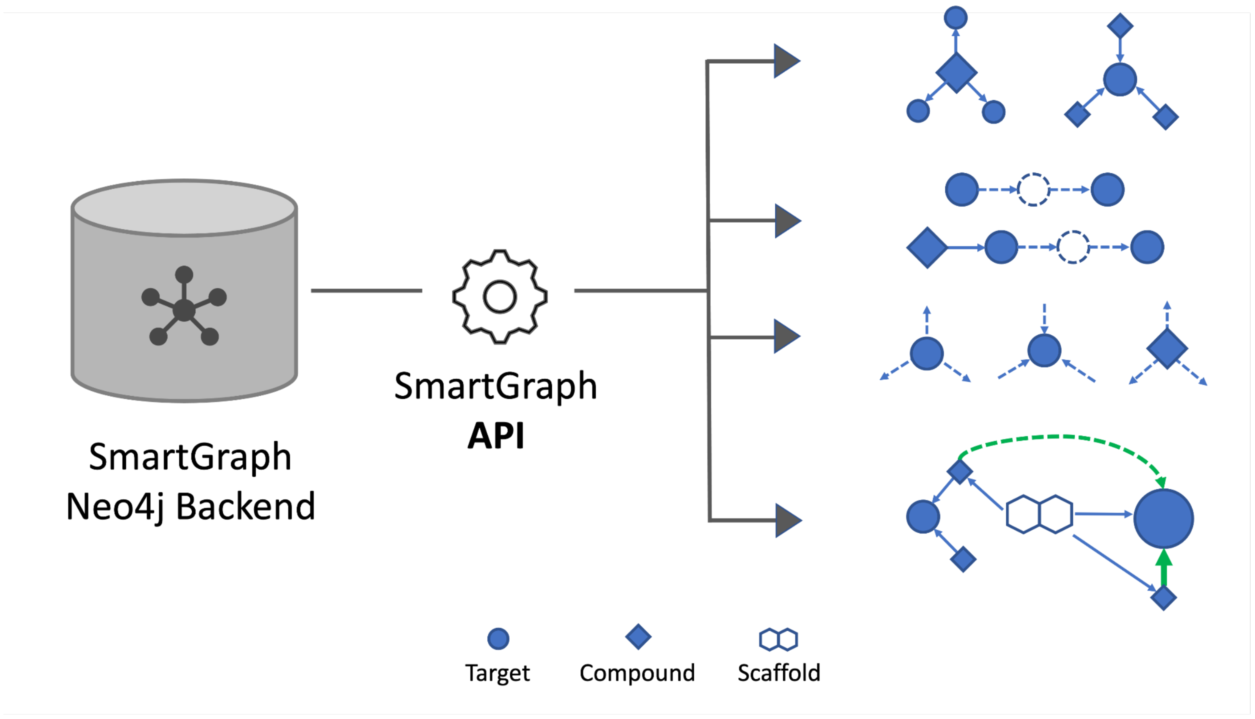


Figure 1. Main functionalities of the SmartGraph API. The figure shows the depiction of the different types of modes to mine and utilize the network-pharmacology knowledge graph with the help of the API. Dashed blue arrows and node outlines indicate the presence or the lack of a path. The enlarged green arrow indicates that the aggregated bioactivity value between the compound and the target exceeds the potency cutoff of the target, hence the compound is a "potent compound" of the target. The dashed green arrow indicates a bioactivity prediction on the basis of the potent pattern of the target (depicted by an enlarged circle).

The above endpoints are examples of "open-ended" queries, where only one of the interacting entity types is known (either target or compound). However, the SmartGraph API supports "Google Maps"-like queries [26], where the source and destination are known. In fact, in the case of most of these types of endpoints, multiple sources and destinations can be defined. An example of this is the "/bioactivity_c2t" endpoint which serves to find bioactivity between a set of compounds and protein targets. This feature is particularly useful if one assumes that a bioactivity value is reported in ChEMBL between a pair of a compound and a protein target. Another two endpoints in this category ("/potent_compounds" and "/potent_patterns") implement logic to find bioactivity relationships utilizing the concepts of "potent compounds" and "potent patterns", as defined in the context of the SmartGraph (see: *Zahoranszky-Kohalmi et al.*, section "Bioactivity prediction" [1]).

The next main category, "Prediction" includes a predictive endpoint ("/predict") that can be used to generate bioactivity predictions building on these concepts. This predictive functionality is particularly useful in a drug-repurposing setting.

The other main categories, "Path search" and "Subgraphs", are focused on extracting paths and subgraphs from the underlying knowledge graph of SmartGraph.
One of the main features of path searches is the ability to limit the length of paths via an argument. In the context of SmartGraph, the rationale behind this is that perturbation in the network is hypothesized to have a more direct effect in "short-range" as opposed to "long-range" given the complex nature of regulatory networks.

Using the API functionality, it is possible to find paths between a set of sources and targets. The targets are proteins, and the sources are either compounds ("/path_c2t") or proteins ("/path_regulatory").

The former endpoint can be used to analyze how a perturbagen, here: a small molecule, might perturb the regulatory pathways. Investigators can adjust the bioactivity cutoff to only take into account compound-target interactions that are more potent than the set value. This feature of the SmartGraph API has direct relevance to studies aimed at elucidating mechanism-of-action in the context of phenotypic screening, as well as designing multi-target therapies.

The latter endpoint might be useful in deriving systems biology-related hypotheses by finding potential mechanistic connections between source and target proteins. To facilitate hypothesis generation, path search can be executed in an "undirected" manner. Accordingly, the direction of regulation is not required to be known a priori of the analysis.

The "/path_regulatory_open" endpoint provides access to an open-ended path search. In this scenario, only the starting points are defined, which are protein targets. Besides controlling the path discovery length, the investigator can define the type of end nodes, *i.e.,* protein targets, compounds, or either. The use of this functionality is relevant to hypothesis development, when only the starting point is known, and one would like to analyze the downstream effects of perturbing a set of protein targets.

Subgraph searches are in close relation to path searches. The focus of these endpoints is to extract subgraphs from the knowledge graph that are induced by a set of compounds ("/subgraph_compound_induced") or protein targets ("/subgraph_target_induced") with the help of a limit on the path lengths. In the latter case, it is possible to explore the network in a directed or undirected manner and to define, whether the provided set of protein targets should be the starting or endpoints of the paths.

Note, that the major difference between "/subgraph_target_induced" and "/path_regulatory_open" is that the latter only allows finding paths between protein target nodes, whereas the former allows paths to start from compounds. An additional difference is that "/path_regulatory_open" allows for controlling the "confidence" property of protein-protein interactions to be included in the resultant paths.

A third endpoint ("/patterns_of_compounds") can be used to extract a subnetwork consisting of compounds and their associated chemical patterns (here: Bemis-Murcko scaffolds [27]).

Case Study

In a recent study by *Zahoranszky-Kohalmi et al.,* we developed and utilized a semi-automated workflow to assemble a COVID-19-focused knowledge graph (Neo4COVID19) involving drug-target and regulatory (protein-protein) interactions. [2] The lack of API for SmartGraph at the time of conducting that study was the reason why the workflow could not be executed in a completely automated manner. The development of SmartGraph API addresses this hiatus. In order to demonstrate the utility of the SmartGraph API, we sought to execute the same workflow described in detail in the Neo4COVID19 study.

The sole difference compared to the original workflow is that the SmartGraph analysis was performed via the SmartGraph API. In the original workflow, we aimed to find potential regulatory connections between a set of "prioritized" histone acetyltransferase proteins (HATs) and other human proteins implicated by pathogen-human protein-protein interactions. Without the API, one needed to submit these two lists into the SmartGraph GUI once as source and end nodes, respectively. However, due to the directed nature of the regulatory relationships, the analysis needs to be performed in the reverse direction as well, for the lack of a priori hypothesis. The resultant networks were required to be saved as JSON files and were subject to further processing. This process is cumbersome and error prone to perform manually, and above all, not scalable.

The SmartGraph API enables us to perform the above steps without manual intervention. Accordingly, the originally pseudo-automated workflow can be executed in a completely automated manner. In the modified workflow, the above described SmartGraph analysis process was performed only by calling the "/path_regulatory" two times.

This endpoint finds directed paths between two sets of nodes, here: HATs and the implicated host proteins. In the first API call the HATs were used as sources and the other set as end nodes, while in the second call the path detection was reversed by swapping the two sets of proteins. The integration of the API endpoint simplified our modified workflow, which was performed by executing only one Python script as opposed to the original three scripts, let alone the manual process of SmartGraph analysis via GUI. The resultant network-pharmacology knowledge graph is nearly identical to that of Neo4COVID19, with minor difference in the number of certain node and edge types (see: *Table S1, S2* in Supporting Information). The difference is accounted for by the changes in the underlying STRING and DrugCentral resources which were likely updated since the publication of the original workflow. Indeed, the set of proteins used as the input of SmartGraph analysis was slightly different as compared to the original Neo4COVID19 study [2], with respect to the HATs and the implicated host proteins as well.

The use of the SmartGraph API resulted in qualitative change as compared to the original workflow. That is, the workflow became more robust, reproducible, and scalable and made it amenable to be integrated into the so-called hypothesis generation engine of the ASPIRE Integrated Computational platform (AICP) [28].

The reproduction of the above workflow is described in detail at :

https://github.com/ncats/neo4covid19/blob/master/README.md

CONCLUSIONS

We introduce the SmartGraph API that addresses a need for the programmatic construction of network-pharmacology-oriented networks, hypothesis generation, and prediction. The API was implemented in the widely utilized FastAPI framework and has been made publicly available at https://github.com/ncats/smartgraph_api_pub . In this study, we provide an overview of the main functionalities of the API. Furthermore, to showcase the change in quality in workflow development and execution, a case study was performed that highlights how the use of the SmartGraph API. We demonstrate this by integrating the SmartGraph API functionalities in the recently published workflow to assemble a COVID-19 an automated workflow, as compared to the original, semi-automated one.

The SmartGraph API and the enclosed case study might further promote the community efforts to facilitate rapid data integration by programmatic means, which is essential for reducing the timeline for finding new treatments for diseases and fighting emerging pathogens.

ASSOCIATED CONTENT

Supporting Information

Process of upgrading the original SmartGraph to use the recent Neo4j v5 database, supplementary tables reflecting the results of the automated workflow enclosed in the "Case Study" section.

Data and Software Availability

The web-based interactive documentation (via Swagger) of the SmartGraph API and the underlying source code are be publicly available at https://smartgraph.scb-ncats.io/api/docs and https://github.com/ncats/smartgraph_api_pub, respectively. The original SmartGraph frontend is available at a new location at https://smartgraph.scb-ncats.io/ui .

The data sets and the workflow utilized in the case study are publicly available at https://github.com/ncats/neo4covid19. Instructions to replicate the workflow can be found here: https://github.com/ncats/neo4covid19/blob/master/README.md.

AUTHOR INFORMATION

Corresponding Author

Gergely Zahoranszky-Kőhalmi, PhD – Email: gergely.zahoranszky-kohalmi@nih.gov

Author Contributions

GZK designed the SmartGraph API created the logic, and performed the implementation of the API endpoints, using the FastAPI framework suggested by HS. BW improved the endpoints by adding support for asynchronous calls, as well as creating the Docker container for the API. NM was leading the architectural development of the containerized SmartGraph platform. NM, BY, DVLP, and SV were pivotal in creating a robust containerized platform for the SmartGraph API and user interface. TIO and TS contributed to the logic of the endpoints. JB suggested upgrading the underlying knowledge graph to Neo4j v5. GZK performed the workflow and wrote the majority of the text. All authors contributed to writing and approve the manuscript.

Funding Sources

This research was supported by the Intramural research program of the NCATS, NIH. This work was supported in part by the Intramural Research Program of the National Center for Advancing Translational Sciences (NCATS), National Institutes of Health, grant number ZIC TR000410-05.

Notes

The authors declare no conflict of interest.

ACKNOWLEDGEMENTS

The authors are thankful for the excellent IT infrastructure support at NCATS/ NIH, especially to Hugo Hernandez and Eduardo Luiggi Lopez.

Supporting Information Available:

Supporting Information: SI_SmartGraph_API.pdf. The file includes:

Sections

- “Upgrading SmartGraph Database to Neo4j v5”

Supplementary Tables

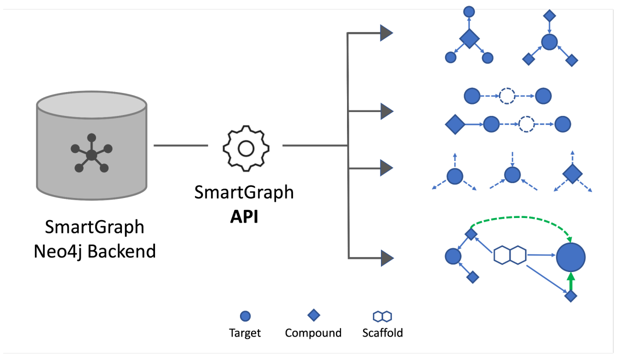
- *Table S1* – *“*Breakdown of the node and edge types according to their data sources in the resultant knowledge graph of the case study.”
- *Table S2* – “ Summary on the various node and edge types in the resultant knowledge graph of the case study.”

REFERENCES

1. Zahoranszky-Kohalmi, G., T. Sheils, and T.I. Oprea, *SmartGraph: a network pharmacology investigation platform.* J Cheminform, 2020. **12**(1): p. 5.
2. Zahoranszky-Kohalmi, G., et al., *A Workflow of Integrated Resources to Catalyze Network Pharmacology Driven COVID-19 Research.* J Chem Inf Model, 2022. **62**(3): p. 718-729.
3. Szklarczyk, D., et al., *STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets.* Nucleic Acids Res, 2019. **47**(D1): p. D607-D613.
4. Ursu, O., et al., *DrugCentral 2018: an update.* Nucleic Acids Res, 2019. **47**(D1): p. D963-D970.
5. Ursu, O., et al., *DrugCentral: online drug compendium.* Nucleic Acids Res, 2017. **45**(D1): p. D932-D939.
6. Gaulton, A., et al., *ChEMBL: a large-scale bioactivity database for drug discovery.* Nucleic Acids Res, 2012. **40**(Database issue): p. D1100-7.
7. Gaulton, A., et al., *The ChEMBL database in 2017.* Nucleic Acids Res, 2017. **45**(D1): p. D945-D954.
8. Kim, S., et al., *PubChem 2023 update.* Nucleic Acids Res, 2023. **51**(D1): p. D1373-D1380.
9. Wang, Y., et al., *PubChem's BioAssay Database.* Nucleic Acids Res, 2012. **40**(Database issue): p. D400-12.
10. Wishart, D.S., et al., *DrugBank 5.0: a major update to the DrugBank database for 2018.* Nucleic Acids Res, 2018. **46**(D1): p. D1074-D1082.
11. Wishart, D.S., et al., *DrugBank: a comprehensive resource for in silico drug discovery and exploration.* Nucleic Acids Res, 2006. **34**(Database issue): p. D668-72.
12. Kanehisa, M. and S. Goto, *KEGG: kyoto encyclopedia of genes and genomes.* Nucleic Acids Res, 2000. **28**(1): p. 27-30.
13. Kelleher, K.J., et al., *Pharos 2023: an integrated resource for the understudied human proteome.* Nucleic Acids Res, 2023. **51**(D1): p. D1405-D1416.
14. Nguyen, D.T., et al., *Pharos: Collating protein information to shed light on the druggable genome.* Nucleic Acids Res, 2017. **45**(D1): p. D995-D1002.
15. Sheils, T., et al., *How to Illuminate the Druggable Genome Using Pharos.* Curr Protoc Bioinformatics, 2020. **69**(1): p. e92.
16. Sheils, T.K., et al., *TCRD and Pharos 2021: mining the human proteome for disease biology.* Nucleic Acids Res, 2021. **49**(D1): p. D1334-D1346.
17. *Ramp DB*. [accessed Nov 24, 2023]; Available from: https://rampdb.nih.gov/api.

18. Braisted, J., et al., *RaMP-DB 2.0: a renovated knowledgebase for deriving biological and chemical insight from metabolites, proteins, and genes.* Bioinformatics, 2023. **39**(1).
19. *Hetionet*. [accessed Nov 24, 2023]; Available from: https://het.io/.
20. Himmelstein, D.S., et al., *Systematic integration of biomedical knowledge prioritizes drugs for repurposing.* Elife, 2017. **6**.
21. *Neo4j Graph Database*. [accessed Nov 24, 2023]; Available from: https://neo4j.com/.
22. *Fast API web application framework*. [accessed Nov 24, 2023]; Available from: https://fastapi.tiangolo.com/.
23. *Python Core Team. Python: A dynamic, open source programming language. Python Software Foundation*. [accessed Nov 24, 2023]; Available from: https://www.python.org/.
24. *Swagger API Documentation*. [accessed Nov 24, 2023]; Available from: https://swagger.io/.
25. *Neo4j driver for Python, v4.4.7*. [accessed Nov 24, 2023]; Available from: https://neo4j.com/docs/api/python-driver/current/
26. *Google Maps*. [accessed Nov 24, 2023]; Available from: https://www.google.com/maps.
27. Bemis, G.W. and M.A. Murcko, *The properties of known drugs. 1. Molecular frameworks.* J Med Chem, 1996. **39**(15): p. 2887-93.
28. *A Specialized Platform for Innovative Research Exploration (ASPIRE)*. [accessed Nov 24, 2023]; Available from: https://ncats.nih.gov/research/research-activities/aspire/laboratory.

For Table of Contents Only


SmartGraph
Neo4j Backend
SmartGraph
API
Target
Compound
Scaffold