Network Query Service

Methods

1. Query

Purpose

Submit queries to the Network Query Service.

Request

Method	URL
GET	http://<hostName>:<portNumber>/graphservice/query?content=value &graph=value&validate=value&view=value&parallel=value&nruns=value

Type	Params	Values	Examples	Required	Description
GET	content	string	• select nodes from netscience where degree > 20 • select edges from netscience where coauthorship> 3 • select sample(5,[3,8]) nodes from dolphins • select edges from sample where u.degree > 1 and v.kshell >1	Yes	Query sting: two forms simple (examples 1 and 2) and sampling (example 3). Example 3 is a new form added in MARS v2.0 which is called mixed queries. Users can query edges based on node attributes, and vice versa.
	graph	string	netscience	Yes	graph name
	validate	string	True/False	No (default False)	Valid entries are True or False only
	view	string	property/seed	Yes	Valid entries are either property (for simple queries) or seed (for sampling queries).
	parallel	string	True/False (Please always use False, feature not supported in the deployed version)	No (default False)	Valid entries are True or False only. If True, sampling queries for seeding will be executed following a thread-based parallelism approach.
	nruns	string	3	No (default0)	Used only if seed view is selected

	details	string	True/False	No (default True)	Valid entries are True or False only. Used to determine whether to return the nodes or edges from a query with or without attributes (IDs only).
	memo	string	True/False	No (default False)	Valid entries are True or False only. If True, query results will be stored in DB for faster response time in future.

Response

Status	Response

200	**Response for simple query will be an object containing the list of nodes or edges as well as the attributes.**
	Note that the WHERE clause in queries is enabled in this version of MARS.
	
	An example response returning selected nodes (33.3% coverage):-
	`select nodes from sample where id < 2 and degree <2`
	
	{"node_sets": [{"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 1, "closeness_centrality": 0.333333333333, "clustering_galib": 0.0, "load_centrality": 0.0, "kshell": 1, "id": 5, "betweeness_centrality": 0.0}, {"clustering": 0.0, "node_clique_number": 2, "degree": 1, "closeness_centrality": 0.333333333333, "clustering_galib": 0.0, "load_centrality": 0.0, "kshell": 1, "id": 6, "betweeness_centrality": 0.0}], "coverage": 33.3}]}
	
	An example response returning all edges in sample network (100% coverage)
	
	`select edges from sample`
	
	{"qid": 2543, "result": {"edge_sets": [{"edges": [{"start": 2, "end": 4}, {"start": 3, "end": 1}, {"start": 3, "end": 5}, {"start": 2, "end": 6}, {"start": 1, "end": 4}], "coverage": 100.0}]}}
	
	New in v2.0: An example response returning all edges in sample network connecting nodes with degree > 1
	
	`select edges from sample where u.degree > 1 and v.degree >1`
	
	{"edge_sets": [{"edges": [{"start": 2, "end": 4}, {"start": 3, "end": 1}, {"start": 1, "end": 4}], "coverage": 60.0}]}

	Response for sampling query. **This query returns three sets of nodes, this is identified by the number "2" after the "number =" keyword. Each set may contain any number of nodes from 1 up to 2, this is shown in "[1,2]" after the keyword "size =".** **Note here that we used a where clause that is acceptable in seed type queries only.** **select sample(number = 2,size =[1,2])nodes from sample where degree > 1** {"node_sets": [{"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.454545454545, "clustering_galib": 0.0, "load_centrality": 0.4, "kshell": 1, "id": 3, "betweeness_centrality": 0.4}], "coverage": 16.7}, {"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.555555555556, "clustering_galib": 0.0, "load_centrality": 0.6, "kshell": 1, "id": 4, "betweeness_centrality": 0.6}], "coverage": 16.7}]} **This query returns three sets of nodes, this is identified by the number "3" after the "number =" keyword. Each set contains exactly one node, this is shown in "[1,1]" after the keyword "size =". Note here that we didn't use the where clause which is optional.** **select sample(number = 3,size =[1,1])nodes from sample** {"node_sets": [{"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.555555555556, "clustering_galib": 0.0, "load_centrality": 0.6, "kshell": 1, "id": 1, "betweeness_centrality": 0.6}], "coverage": 16.7}, {"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 1, "closeness_centrality": 0.333333333333, "clustering_galib": 0.0, "load_centrality": 0.0, "kshell": 1, "id": 5, "betweeness_centrality": 0.0}], "coverage": 16.7}, {"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.555555555556, "clustering_galib": 0.0, "load_centrality": 0.6, "kshell": 1, "id": 4, "betweeness_centrality": 0.6}], "coverage": 16.7}]}

	New in v2.0 **Response for a subgraph query.** **This query returns a subgraph of nodes, starting node is identified by the number "4". Number of levels is identified by the second number "1". This is a special form of subgraph using breadth-first search. This query returns a node with all its distance-1 neighbors.** **Note: this approach is working with small-medium networks. Need to be optimized for better performance with larger networks. Other algorithms like depth-first can be added.** **select subgraph(4,1) from sample** {"node_sets": [{"nodes": [{"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.555555555556, "clustering_galib": 0.0, "load_centrality": 0.6, "kshell": 1, "id": 1, "betweeness_centrality": 0.6}, {"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.454545454545, "clustering_galib": 0.0, "load_centrality": 0.4, "kshell": 1, "id": 2, "betweeness_centrality": 0.4}, {"clustering": 0.0, "node_clique_number": 2, "degree": 2, "closeness_centrality": 0.555555555556, "clustering_galib": 0.0, "load_centrality": 0.6, "kshell": 1, "id": 4, "betweeness_centrality": 0.6}], "coverage": 50.0}]} New in v2.0 **Response for a compound query. This query involves execution of multiple queries (two or more) and the results are combined using union, intersect, or except.** **select nodes from sample where load_centrality < 0.4** **intersect select nodes from sample where id > 5** {"node_sets": [{"nodes": [{"id": 6}], "coverage": 16.7}]}
200	**Response for query validation** {"check": "Valid"}
400	"Invalid Query"
300	"Do you mean degree" if user misspelled attribute name, service responds with the closest attribute suggestion

2. Set operations

Request

Method	URL	Description
GET	http://<hostName>:<portNumber>/graphservice/session/create?appid=value	This end point creates a session for an application with id identified by appid. The end point returns the session id.
GET	http://<hostName>:<portNumber>/graphservice/session/end?appid=value&sessionsid=value	This end point ends an existing session with id equals to sessionid for an application with id equals to appid.
GET	http://<hostName>:<portNumber>/graphservice/setoperation/delete?setid=value	This end point deletes an existing set identified by id equals to setid.
GET	http://<hostName>:<portNumber>/graphservice/setoperation/save?setname=value&sessionid=value&queryid=value	This end point saves a query (identified by queryid) result into a set with name setname. The set will belong to session with id equals to sessionid.

Network Query Search Service

1. Search

Purpose

Search for existing queries. For example, in the EDISON web application, this invocation is used to return all queries that have been executed, so that they may be displayed in the UI, to help users form their own queries or use pre-existing ones.

Request

Method	URL
GET	http://<hostName>:<portNumber>/graphservice/search?keywords=value&metadata=value&view=value

Type	Params	Values	Examples (Valid entries)	Required	Description
GET	keywords	String	sample NOT degree	yes	Keywords used for query searching. Users can use Boolean operators as an example AND, OR, NOT
	metadata	string	content/graph/query_id/target	No (Default content)	Type of metadata searching is based on. Valid entries are only content, graph, query_id or target. “query_id” is an integer number that uniquely identifies the query. “target” is the target type which can be node or edge. “graph” is the name of the network on which the query is executed. “content” is the query string constructed by end user.
	view	string	property/seed	yes	The type of queries. Valid entries are property or seed only.

Response

Status	Response
200	**Response will be an object containing the list of queries (array).** **Using the keywords sample NOT degree on metadata content which means search for all the queries with contents that have a keyword sample but not degree.** **Note: the Boolean operation (e.g. NOT, OR) shall be all capital, use as many operators as you want** [{"content": "select edges from sample", "graph": "sample", "query_id": "2530", "results": "/home/sipcinet/edison/graphservices/query/sample_query2530.txt", "target": "edge"}, {"content": "select nodes from sample", "graph": "sample", "query_id": "2527", "results": "/home/sipcinet/edison/graphservices/query/sample_query2527.txt", "target": "node"}]
400	"Error"

2. Repository

Purpose

Repository keeps information about all queries. Categorize queries by node and edge.

Request

Method	URL
GET	http://<hostName>:<portNumber>/graphservice/repository?type=value&view=value

Type	Params	Values	Examples	Required	Description
GET	type	string	node/edge/ all	yes	Category of data to be retrieved. Valid entries are node, edge or all only
	view	string	property/seed	yes	Type of queries. Valid entries are property or seed only.

Response

Status	Response
200	**An example response for choosing type node and view seed queries. This will return all queries targeting nodes of type seed for sampling.** {"node_queries": [{"content": "select sample(number = 3,size =[1,1])nodes from sample", "graph": "sample", "query_id": "2532", "results": "/home/sipcinet/edison/graphservices/query/sample_query2532.txt", "target": "node"}, {"content": "select sample(number = 2,size =[1,2])nodes from sample where degree > 1", "graph": "sample", "query_id": "2531", "results": "/home/sipcinet/edison/graphservices/query/sample_query2531.txt", "target": "node"}]} **An example response for choosing type node and view property queries. This will return all queries targeting nodes of type simple.** {"node_queries": [{"content": "select nodes from sample where degree = 1", "graph": "sample", "query_id": "2529", "results": "/home/sipcinet/edison/graphservices/query/sample_query2529.txt", "target": "node"}, {"content": "select nodes from sample where degree >2" "graph": "sample", "query_id": "2528", "results": "/home/sipcinet/edison/graphservices/query/sample_query2528.txt", "target": "node"}, {"content": "select nodes from sample", "graph": "sample", "query_id": "2527", "results": "/home/sipcinet/edison/graphservices/query/sample_query2527.txt", "target": "node"}]}
400	"Error"

Network Storage Service

Methods

1. Graph

Purpose

List all stored graphs. Search a graph by name or filter graphs by attribute value

Request

ID	Method	URL	Description
1	GET	http://<hostName>:<portNumber>/graphservice/storage/graph	This end point returns all graphs marked as available
2	GET	http://<hostName>:<portNumber>/graphservice/storage/graph/filter?attribute=value&operator=value&rval=value	This end point returns all graphs marked as available and satisfy the given filter. Filter is defined by attribute, operator and value
3	GET	http://<hostName>:<portNumber>/graphservice/storage/measure_notify?graph=value&measure=value	This end point is used by the measure service to notify storage service that a requested measure computation is complete.

Type	Params	Values	Example	Description
GET	attribute	string	network attribute	Network attributed used for filtering. Network attributes are part of network metadata.
	operator	string	>,>=, <, <=, !=,=	Operator used for the comparison. Valid values are >,>=, <, <=, !=,=
	rvalue	string	500	Right-hand value. This can be string or number. Has to be consistent with the attribute data type. For example, if the attribute on left-hand side is of type string, the rvalue shall be string too.

Response

Status	Response
200	**Response for method 1 and 2 will be a JSON object containing a single graph or a list of graphs with details, including title, description and other metadata. Here we sent a request using attribute nodes, operator <, and rvalue 40. This will return all graphs in repository that are marked available to public and has number of nodes < 40.** [{"directed": "false", "weighted": "false", "graph_id": 38, "name": "karate", "edge_attributes": {"degree_product": "integer", "betweenness_centrality": "real"}, "numberOfEdges": 78, "file_name": "karate", "original_format": "uel", "labeled": "true" "node_attributes": {"node_clique_number": "integer", "closeness_centrality": "real", "degree": "integer", "betweenness_centrality": "real", "load_centrality": "real", "id": "integer", "clustering": "real"}, "numberOfNodes": 34, "description": "Network of friendships between the 34 members of a karate club at a US university, as describe by Wayne Zachary in 1977"}]
400	"Error"

2. ADD Network

Note: This feature is disabled for the current EDISON version. EDISON has no screen that enables users to upload networks. However, this feature can be called from Rest API directly by admin users.

Request

Method	URL
GET	http://<hostName>:<portNumber>/graphservice/storage/addnetwork

Type	Params	Values	Example	Description
GET	name	string	karate, netscience	Name of graph to be added. Graph files (.uel, .nodes, and .md) need to be placed manually or uploaded (through UI) before calling this method. Currently, there is no checking for graph validity. This is the responsibility of admin users of mars. The upload directory is the uploadfile property defined in mars.config file. If uploading is going to be done through the web app UI, then the upload directory should be accessible by the app. • .md: has the graph metadata • .uel: list of graph edges • .nodes: list of graph nodes

Response

Status	Response
200	OK - graph added successfully
400	"Error"

Network Measure Service

Methods

1. Compute

Purpose

Submit measure computation requests to the measure service.

Request

Method	URL
GET	http://<hostName>:<portNumber>/graphservice/measure/compute?graph=value &measure=value

Type	Params	Values	Examples	Required	Description
GET	graph	string	karate, dolphins, lesmis	Yes	Graph name
	measure	string	measure id	Yes	A number that uniquely identify the measure. 1. degree 2. betweeness_centrality 3. clustering 4. load_centrality 5. node_clique_number 6. closeness_centrality 7. clustering_galib 8. kshell These are all the valid values. New measures/ids can be added in the future.

Response

Status	Response

200	OK - measure computed successfully
400	“Error”

User-defined Workflow Service

Methods

Purpose

Submit user-defined workflow execution requests to the workflow service.

Request

Method	URL
GET	http://<hostName>:<portNumber>/workflowservice/execute?wf=value &input=value

Type	Params	Values	Examples	Required	Description
GET	wf	string	• start,network_data,min_data,end • start,execute_query,query_data,sum_data,end	Yes	Sequence of processes that to be executed. Processes names are separated by “,”. The expected input/output data type of consecutive processes shall be compatible.

Process	Functionality
network_data	Extracts node or edge attribute data. A filter can be applied.
start	Start the execution session of the workflow and prepare input data for each process.
end	End the execution session of the workflow. Return the final results.
plot_data	Plot data (e.g. degree distribution, clustering distribution)
save_data	Saves output data from a process into a variable. Currently, saves a single value, but can be extended to store multiple values as a list.

					execute_query	Send a network query request to query service.
					query_data	Parse and extract output data (JSON response) of the execute_query process
					min_data max_data average_data product_data std_data var_data median_data percentile_data	These are group of processes that statically analyze network data (e.g. min, max, max, count, average, median). They are using numpy library but new packages can be added in the future.
	input	string	• degree,sample,node,no condition • select nodes from sample where degree =1 ,sample,node|clustering	Yes	Input data for each process (if needed). Some processes don't need input.	

Response

Status	Response
200	OK - workflow executed successfully
400	"Error"

Model Information Service

Note: This service is in not use now, and will take over in the future all model-related processes from Edison front end. Currently, returns static data for demo purpose. New functionality will be added.

Methods

Purpose

List all Edison models with information

Request

Method	URL
GET	http://<hostName>:<portNumber>/graphservice/model

Response

Status	Response
200	**Response will be a JSON object containing the a list of models with details, including description, parameters and other metadata** [{"sub_model_id": 1, "model_id": 11, "model_name": "Progressive 2-state model", "parameters": [{"threshold": "integer", "model": "integer", "type": "int_node_trait", "sub_model": "integer"}, {"state": "integer", "is_fixed": "integer", "type": "int_node_state"}], "description": "Threshold model where nodes may transition from state 0 t 1, but not from 1 to 0. The model supports blocking nodes: nodes that do not change state."}, {"sub_model_id": 3, "model_id": 11, "model_name": "Back-and-forth 2-state model", "parameters": [{"type": "int_node_trait", "model": "integer", "down_threshold": "integer", "sub_model": "integer", "up_threshold": "integer"}, {"state": "integer" "is_fixed": "integer", "type": "int_node_state"}], "description": "Threshold model where nodes may transition from state 0 to 1, and from 1 to 0. The model supports blocking nodes: nodes that do not change state."}, {"sub_model_id": 4, "model_id": 11, "model_name": "Back-and-forth 2-state model with influence from distance-2 neighbors", "parameters": [{"type": "int_node_trait", "model": "integer", "down_threshold": "integer", "sub_model": "integer", "up_threshold": "integer"}, {"state": "integer", "is_fixed": "integer", "type": "int_node_state"}], "description": "Threshold model where nodes may transition from state 0 to 1, and from 1 to 0. Neighboring nodes at distance 1 and distance 2 can influence a node."}, {"sub_model_id": 1, "parameters": [{"model": "integer", "duration_in_state_I": "integer", "type": "int_node_trait", "sub_model": "integer"},

	{"state": "integer", "type": "int_node_state"}, {"edge_weight": "double", "type": "double_edge_state"}], "model_name": "SIR epidemic model", "model id": 22, "description": "Classic susceptible-infected-recovered epidemiological model."}, {"sub_model_id": 3, "model_id": 22, "model_name": "SIR epidemic model", "parameters": [{"model": "integer", "type": "int_node_trait", "sub_model": "integer", "duration_in_state I": "integer"}, {"threshold": "integer", "state": "integer", "type": "int_node_state"}, {"edge weight": "double", "type": "double_edge_state"}], "description": "Classic susceptible infected-recovered epidemiological model, but a susceptibl node may require multiple infecting neighbors to become infected."}, {"sub_model_id": 0, "model_id": 37, "model_name": "Linear Threshold model", "parameters": [{"model": "integer", "type": "int_node_trait", "sub_model": "integer"}, {"state": "integer", "type": "int_node_state"}, {"threshold": "double", "type": "double_node_state"}, {"type": "double_edge_state", "edge influence": "double"}], "description": "This is the Linear Threshold model of Kempe et. al (KDD 2003)."}, {"sub_model_id": 0, "parameters": [{"model": "integer", "type": "int_node_trait", "sub_model": "integer"}, {"state": "integer", "is_fixed": "integer", "type": "int_node_state", "down_threshold": "integer", "up_threshold": "integer"}], "model_name": "Connected Components Threshold Model", "model id": 38, "description" "This is an influence model that uses thresholds, but now each set of neighbors of a node that form a connected component collectively influence the node (Ugander, PNAS 2012)."}]