GOLEM: Flexible Evolutionary Design of Graph Representations of Physical and Digital Objects

Maiia Pinchuk*
Grigorii Kirgizov*
Lyubov Yamshchikova*
Nikolay Nikitin
maiiapinchuk@gmail.com
gvkirgizov@itmo.ru
laiamshchikova@itmo.ru
nnikitin@itmo.ru
ITMO University
Saint-Petersburg, Russia

Irina Deeva
Karine Shakhkyan
Ivan Borisov
Kirill Zharkov
Anna Kalyuzhnaya
ideeva@itmo.ru
kashakhkyan@itmo.ru
borisovii@itmo.ru
kdzharkov@itmo.ru
anna.kalyuzhnaya@itmo.ru
ITMO University
Saint-Petersburg, Russia

ABSTRACT

We introduce GOLEM — an open-source optimization framework for automated design of graph-based structures in various scientific domains. It solves the problem of finding optimal topology and parameters of graphs using evolutionary algorithms, and does it in a modular domain-agnostic way. The paper describes the framework and its flexible approach to domain adaptation. Experimental studies provide several examples of GOLEM application in different fields: Bayesian networks, drug design, and robotics.

CCS CONCEPTS

• **Computing methodologies**; • **Software and its engineering**;

KEYWORDS

Evolutionary Design, Graph Search, Open Source, Robotics, Drug Design

ACM Reference Format:
Maiia Pinchuk, Grigorii Kirgizov, Lyubov Yamshchikova, Nikolay Nikitin, Irina Deeva, Karine Shakhkyan, Ivan Borisov, Kirill Zharkov, and Anna Kalyuzhnaya. 2024. GOLEM: Flexible Evolutionary Design of Graph Representations of Physical and Digital Objects. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3638530.3664141

*These authors contributed equally to this research.

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07
https://doi.org/10.1145/3638530.3664141

1 INTRODUCTION

The problem of finding optimal structures in the form of graphs arises in many scientific and industrial domains [35]. Examples range from probabilistic models design, as in Bayesian Network (BN) modeling [24], Neural Architecture Search (NAS) [9], and automatic design of machine learning pipelines (AutoML) [15], to real-world applications, such as design of engineering structures [40], robotic skeletons [31], and molecular structures in drug design [30]. These types of problems require deep domain expertise and many rounds of prototyping and experiments, which can take considerable resources. So, there is need for instrument which can automate graph design tasks.

We describe this class of tasks as *graph search and optimization* [6]. It involves finding the optimal topology (search) and parameters of the nodes (optimization). Characteristics of this task include: discrete problem space, small to medium graph size (from 10 to 100 nodes), sometimes with highly heterogeneous nodes requiring many differing parameters (e.g. AutoML where each node is ML model with its own set of hyperparameters), and often lacking solution datasets.

This class is a part of the broader *graph learning* field, which includes such tasks as graph, node, and link classification and prediction. While the topic of graph learning have acquired a lot of attention in recent years, especially with Deep Learning and Graph Neural Networks (GNN), and have made impressive progress in many applications, e.g. drug design [21], GNN methods are often not directly applicable to such tasks. In particular, AutoML field is still dominated by meta-heuristic methods [15], and deep neural methods show controversial results [16].

Meta-heuristic methods [1], such as evolutionary search or particle swarm optimization, represent one of the approaches to graph learning problems. These methods don't require gradients and can work directly on discrete problem spaces. Decades of experience of using these methods show that it can achieve good results in complex search spaces [17] showing comparable or even better results than gradient-based neural network methods [27, 37]. All of the above makes it an excellent fit for graph optimization, where solution space has discrete nature.

To our knowledge, there is no accessible and universal tool for graph search and optimization despite the generic problem formulation. But why there is no such tool? On the one hand, most of the effective methods proposed in the literature are task- or domain-specific and are difficult to generalize [23, 46] On the other hand, existing general-purpose optimisation frameworks, both gradient-based and meta-heuristic [2] can not be applied to graphs without either large dataset of known solutions, which are rarely available in specific domains, or extensive integration work with meta-heuristic frameworks which are not designed for graphs.

Our contribution attempts to close this gap. We introduce GOLEM: a domain-agnostic framework for graph optimization with evolutionary methods. GOLEM supports discrete graph optimization natively, implements graph-specific mutations and supporting infrastructure, significantly lowering the required effort for using it in a new domain. GOLEM is an open-source project, available on GitHub[1] under BSD-3 license. In the paper we describe successful applications of GOLEM in several domains: probabilistic modelling, drug design, robotics, and compare them with baselines.

2 RELATED WORKS

We have divided the existing approaches in graph learning into three groups: general-purpose optimisation frameworks, methods that are based on Graph Neural Network (GNN), and domain applications.

2.1 General-purpose frameworks for graph optimization

There are several meta-heuristic frameworks in the field, ranging from research endeavours to production-ready libraries. DEAP is a popular framework for fast prototyping of evolutionary algorithms. Two other popular examples are Pymoo [2], a library for real-valued multi-objective optimization, and PyPop7 [8], which is a "Pure-PYthon library of POPulation-based OPtimization for single-objective, real-parameter, black-box problems". They include a diverse set of black-box optimization methods, including evolutionary algorithms.

Another category is gradient-based evolutionary frameworks like EvoTorch or EvoJAX based on PyTorch and JAX frameworks respectively. While its primary application is neuroevolution, it can be used for optimization in any real-valued domain with an objective.

The existing meta-heuristic frameworks require a bridge into graph optimisation domain. For example, in theory, DEAP can be used for optimization of any discrete structure. However, it requires considerable implementation effort, as DEAP provides only basic infrastructure for evolution. It can be trivial to map a graph into its adjacency matrix to enable optimization with these frameworks, but this task becomes more complicated for parameterized graphs or graphs with non-fixed size.

2.2 Graph Neural Networks

Graph Neural Network (GNN) can be considered as efficient tool for graph design. GNN-based methods are implemented in libraries like Jraph [11], PyG [10], DGL [45] and others. It shows state-of-the-art results on a large number of graph-related problems (e.g. optimal power flow [34], drug discovery [20], NAS [33]). However, main tasks for GNN are graph, node and edge predictions. There are several GNN-based methods exist that aimed specifically at graph search or generation: both domain-specific (e.g. drug discovery [28]) and domain-agnostic (e.g. approaches for labeled graphs [12], solutions based at hierarchical RNN [49], random walks algorithm, score-based [32] and multi-scale [51] generative models.

However, GNN require large datasets of known solutions, which are rather rarely available in potential application domains. Even if suitable data exists, it can introduce significant bias [27]. Meta-heuristic methods, on the other hand, do not require datasets for learning, and only need an objective function, or, at most, a handful of exemplary solutions to compare against. GNN methods are also less applicable for smaller graphs (from 10 to 100 nodes) and highly heterogeneous graphs, mostly because of data insufficiency in such cases for generalisation of node and graph embeddings.

2.3 Graph optimization in specific domains

There are numerous applications in multiple domains aimed at graph optimization. We highlight only ones that are relevant for the case studies considered in the paper.

One example is G2o: a general framework for graph optimization in the field of robotics and simultaneous localization and mapping (SLAM). Its main idea starts from the observation, that many popular problems in robotics and computer vision including various types of SLAM or bundle adjustment (BA) can be formulated as least squares optimization of an error function that can be represented by a graph [13].

Graph optimization encompasses the structural learning of BN, particularly when dataset assumes the form of a graph, where nodes correspond to features, and edges to dependencies between them. The graph serves as a model of a multivariate distribution that characterizes the data. Classical BN learning algorithms use conditional independence tests (CI) [19]. In this direction, the most modern solution is the *PC* algorithm [5]. A number of learning algorithms is based on the formulation of the task as an optimization problem in the space of possible structures with an evaluation function. Such algorithms include the K2 algorithm, Hill-Climbing, Sparse-Candidate. There are also hybrid approaches that combine the CI and score-based algorithms [43]. These algorithms are specific for learning BN and can not be generalized to different scenarios, such as nodes containing observations instead of features, or optimization criteria different from quality measures for multivariate distributions. Finally, the involvement of continuous optimisation into BN learning provides the promising results [50].

Goal-directed molecule generation can be considered as graph structure optimisation problem [27]. Although string representation using SMILES [47] is also quite popular [39], some debate the efficiency of this approach since symbol changes often result in invalid structures [18]. Existing solutions can be classified to algorithms involving some of the latest advances in deep learning [44], and classical optimisation techniques, for instance, evolutionary

[1] https://github.com/aimclub/GOLEM

algorithms. However, machine learning based solutions rely heavily on the training data, so such models can be biased to known molecular structures, which affects their exploration capabilities.

3 PROBLEM STATEMENT

There are various formulations of graph learning tasks in literature [22, 35]. The problem solved by GOLEM is minimization of the objective F in the discrete space of parameterized directed graphs $\mathbb{G}$ subject to a set of arbitrary graph constraints $\mathbb{C}$. The task is to find optimal graph $G^* = \langle V,\ E,\ P_V \rangle$ with a set of vertex parameters P_V.

$$G^* = \underset{\mathbb{G}}{argmin}\ F\ where\ \mathbb{G} = \{G \mid \mathbb{C}(G)\ is\ True\}$$

In multi-objective case the minimization finds Pareto-front of optimal solutions. Any domain task can be reduced to this optimization task as long as we can define a mapping A (where A stands for Adapter) from the space of domain structures $\mathbb{S}$ into parameterized directed graphs together with its reverse mapping R (where R stands for Restore). Its role will be explicated further.

$$A : \mathbb{S} \rightarrow \mathbb{G}\ and\ R : \mathbb{G} \rightarrow \mathbb{S}$$

4 PROPOSED APPROACH

In this section we describe the implemented approach to the stated problem. The GOLEM framework consists of three logical layers:

- *Optimization core* that's responsible for operation with graphs, optimization algorithms, objectives, stop conditions etc. It is domain-independent and works on internal graph representation.
- *Domain specification layer* that's responsible for bidirectional transformation between domain structures and internal graphs. It is a bridge between a specific domain and universal optimization core.
- *Infrastructural layer* that's responsible for non-core functionality such as serialization and visualisation and technical details of evaluation such as caching and parallelization. It brings usability, effectiveness, interpretability and reproducibility of experiments.

4.1 Optimization core

Optimization core is a set of abstractions related to: graphs, graph optimizers, objective functions, adaptive stopping conditions and specific implementations of optimization algorithms. The aim of this module is to solve the problem defined in Section 2.3.

Primary optimization algorithm is an implementation of multi-objective evolutionary algorithm based on SPEA-2 selection with mutation and crossover operators designed for graphs. The choice of well-known SPEA-2 is inspired by its successfully application in various graph-related applications [40]. SPEA-2 selection is used by default, however, there is a possibility to use custom selection algorithm or to choose from the ones implemented in GOLEM. Also, we add a verification step to accommodate custom domain-specific constraints on graphs.

It is also often a case that the objective is a partial function that can't be evaluated for some individuals, for example simulation-based objectives with timeout. If there're too many individuals that cannot be evaluated, performance of the evolutionary algorithm can degrade due to decreasing and unpredictable population size. To resolve this practical issue, we add a *Reproduction Controller* to the algorithm, that implements proportional controller for population size to compensate for invalid evaluations.

Figure 1 presents a scheme of the optimization core with evolutionary algorithm integrated. Input parameters of the optimization algorithm include graph search space (available node types and node parameters), objective function and initial population of graphs. Also, is can optionally include constraints and custom operators. When evolutionary operators are used (mutations, crossovers), selection is applied and graph constraint rules are considered at each iteration on the population of individual solutions. Reproduction controller keeps population size stable. After evolution the hyperparameters of the best graphs (parameters of graph nodes) are optimized with the external tuner.

Mutations include basic single-point graph modifications: adding, removing, or exchanging a randomly chosen node or edge. Also, the complex growth and reducing mutations are implemented for faster exploration of the search space. For instance, a new subgraph can be added as a mutation. Single-point graph modifications make mutation set more universal, while more complex growth and reduce mutations enable faster exploration of the search space.

4.2 Domain adaptation layer

For the definition of the new domain task it's required to specify the following mappings and parameters, according to the task definition in Section 2.3.

- $\mathbb{G}$: Search space of graphs with parameterized nodes.
- F: Objective function to be minimized.
- A and R: Mappings between domain structures and graph representation.
- $\mathbb{C}$: Set of constraints (optional).

Primary feature of GOLEM is universality. Any domain structure that can be converted to parameterized graphs can be optimized with GOLEM. It provides flexibility in definition of domain structures.

Adapt and Restore mappings enable natural definition in domain terms of the objectives, custom evolutionary operators and custom structure constraints. GOLEM applies these mappings to adapt provided domain cost functions and domain operators to graph representations. For example, if there is a domain objective F defined on the space of domain structures $F : \mathbb{S} \rightarrow \mathbb{R}$, an objective F' defined for graphs can be obtained: $F \circ A = F' : \mathbb{G} \rightarrow \mathbb{R}$. Similarly, if there is a domain mutation $M : \mathbb{S} \rightarrow \mathbb{S}$, a graph mutation $R \circ M \circ A = M' : \mathbb{G} \rightarrow \mathbb{G}$ can be obtained. Both F' and M' after described transformations can be used in universal graph optimizer. The same logic works for constraints on domain structures. Most importantly, these transformations are applied automatically by GOLEM. Users are only required to provide Adapt and Restore mappings by implementing `GraphAdapter` interface with corresponding methods `adapt` and `restore`.

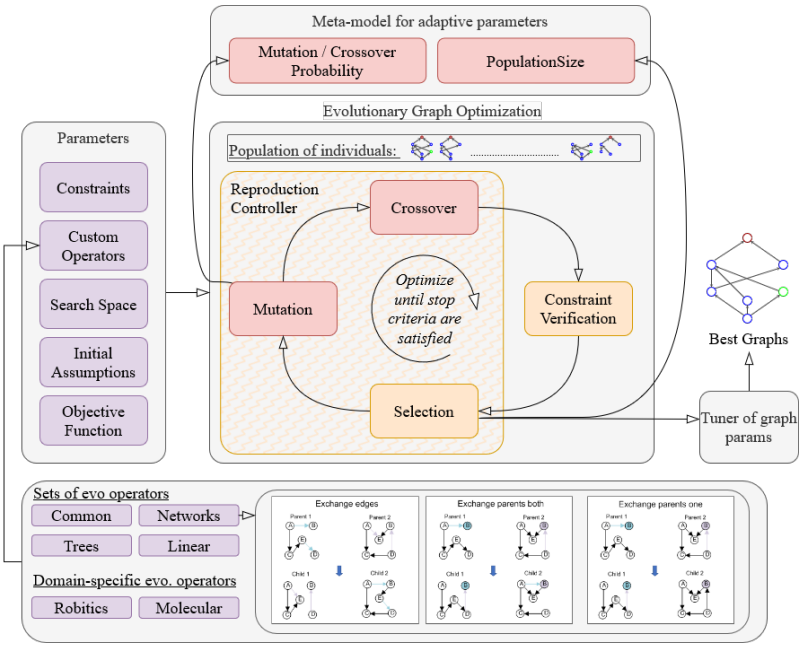

Figure 1: The scheme of the GOLEM framework: main blocks and stages

5 CASES AND EXPERIMENTAL STUDIES

Most of the existing benchmarks related to graph learning consider exclusively graph, node, and link prediction tasks (e.g. [29]), and do not include graph search tasks (in its current form). One of the few examples of suitable solutions is single-task drug design benchmark [4] and more diverse model-based design benchmark [42]. We combined ideas from the existing benchmarks and designed the setup of experiments on synthetic graph search tasks and real-world tasks in specific domains.

5.1 Synthetic experiments

For synthetic experiments tree and DAG types of graphs were used. To generate tree graphs `random_tree` function from NetworkX is used, however, for DAG graphs generation `gnp_random_graph` is used first and then the result graph is postprocessed by removing edges until it becomes directed and acyclic.

Metrics for graph comparison were used in according with analysis in [48]. For this set of experiments, the following metrics that characterize topology of graphs were used:

- `degree_dist` is a heuristic metric for graphs where central nodes are more important than peripheral ones. The higher the nodes degree, the more significant the difference in number of such nodes between two graphs. Thus, degree histogram is calculated for each graph and then distance between these histograms is computed. This metric characterizes local topology of a graph.
- `sp_adj` is characterizes local and global topology of a graph and is calculated as L2 distance between spectres of adjacency matrices of graphs.

Table 1: Comparison of restoration quality and convergence time for random search (RS) baseline and evolutionary optimizer in GOLEM. The best results are highlighted with bold.

metric	degree		graph_size		sp_adj	
	Restoration error					
target	RS	Evo	RS	Evo	RS	Evo
d_50	0.72	**0.70**	24.72	**23.01**	4.94	**4.60**
d_100	**0.62**	0.65	94.55	**75.64**	18.91	**15.13**
t_50	0.37	**0.35**	10.49	**3.47**	2.10	**0.69**
t_100	**0.59**	0.80	51.95	**25.95**	10.39	**5.19**

- `graph_size` is the difference between sizes of the graphs. It is included in experiments as a simplest baseline metric.

The main difference between compared optimizers is that random optimizer iteratively applies randomly chooses mutation for each graph in population, while evolutionary optimizer also uses evolutionary selection based on SPEA2 and one-point crossover for each pair of individuals.

Each experiment was launched 15 times for 350 generations. For each graph type, there were 50 and 100 target graph sizes. The results are presented in Table 1. Evolutionary optimizer significantly outperforms random one for the vast majority of setups. Furthermore, to find out the statistical significance of the obtained results, a non-parametric Mann-Whitney test was used and it was confirmed that for almost all setups the results are statistically significant.

The examples of convergence plots for `sp_adj` metric and tree graph types of size 100 are presented in Figure 2a. It can be seen that in addition to the fact that the evolutionary optimizer outperforms the random one in terms of metrics, it also converges at the

early stages of optimization, which makes it more optimal even on launches with fewer generations.

The main advantage of GOLEM framework is that it is applicable to a variety of domains. So, for example, it is used as the core of the AutoML framework FEDOT[2]. However, in machine learning, in most cases the optimal solution lies among small pipelines consisting of ML models and data preprocessing operations. The full potential of the GOLEM is revealed in the real-world specific tasks presented below.

5.2 Bayesian networks

During the analysis of GOLEM performance, we turn our attention to the task of Bayesian network (BN) structural learning. The field of evolutionary algorithms has seen notable application in the context of BN structure learning [25]. Nevertheless, it is noteworthy that prevailing approaches have primarily focused on optimizing not the underlying graphical structure per se, but rather its encoded representations, such as the adjacency matrices [14], ordered node lists [26], among others. Distinctively, the presented GOLEM-based solution introduces a novel approach, facilitating evolutionary optimization directly targeting the BN's structural configuration.

The implementation of structural learning for Bayesian networks (BN) entailed the integration of two distinct frameworks: BAMT[3], which specializes in working with Bayesian networks[7], and the GOLEM framework. Specifically, within the context of the BAMT framework, we defined the requisite mutation and crossover operations, the evaluation function, and the guidelines governing the selection of valid structural configurations. The K2 function was used as an evaluation function, which has a fairly wide application in structural learning problems.

To facilitate a comparative analysis of structured learning outcomes, a series of experiments were conducted on the designated benchmark datasets [38]. As the foundational baseline, the Greedy Hill Climbing algorithm was selected. The learning outcomes are assessed from a dual perspective, utilizing two distinct metrics: F1 score and the evaluation function's value. In this case, F1 score shows the quality of the restoration of the structure (the higher, the more similar the structure is to the reference one). Table 2 shows the comparison results (averaged for ten runs). The comparative analysis revealed that the evolutionary optimization algorithm integrated into the GOLEM framework effectively addresses the challenge of structural training for Bayesian networks, yielding superior outcomes when contrasted with the baseline algorithm. As the number of nodes increases, a comparable result is achieved. However further refinement is required on a large number of nodes to exceed the foundational baseline.

Figure 2b illustrates the convergence plots of solutions from ten runs, employing the Sachs dataset as an illustrative case. Notably, even though the solutions ultimately converged to identical values of the evaluation function, it is worth highlighting that GOLEM managed to uncover a superior solution when assessed through the lens of the F1 metric. This phenomenon arises from the notion of equivalence within the domain of Bayesian networks. Namely, different network structures can correspond to identical evaluation function values. Given that evolutionary algorithms inherently explore a broader spectrum of solutions, it possesses a higher likelihood of converging towards the desired reference structure. In contrast, the baseline solution typically exhibits a more deterministic convergence pattern towards a consistent structure.

Table 2: Comparison of BN's structural learning results for evolutionary algorithm (Evo) and the baseline - greedy algorithm (HC), for each dataset, the number of nodes is indicated in brackets. The best results are highlighted with bold.

Dataset	Evo	HC	Evo	HC
	F1		*Evaluation function*	
Asia (8)	**0.71**	0.29	**-2296**	-2300
Cancer (5)	**0.57**	0.57	**-2104**	-2105
Earthquake (5)	**1.00**	1.00	**-533**	-533
Sachs (11)	**0.62**	0.53	-7400	**-7391**
Sangiovese (15)	**0.2**	0.17	**-22835**	-22878
Mildew (35)	**0.28**	0.28	-52749	**-52076**
Barley (48)	**0.24**	0.23	-63835	**-61448**

5.3 Drug design

As another case to demonstrate GOLEM's versatility and effectiveness we have chosen the drug discovery task. Its purpose is to discover molecules with specific chemical properties. Exploring the vast molecular space is a challenging task, primary due to its extensive size. To tackle the problem of goal-directed molecule generation, we applied GOLEM framework using RDKit chemical library.

To obtain new molecular structures we used custom mutations proposed in [27]. Implemented mutation operators can be divided into two groups: primary and secondary. Primary mutations are simple operations like adding, deleting or replacing an atom and deleting or replacing a bond. More complicated, multi-step actions correspond to secondary mutations: deleting or moving a functional group, inserting carbon between two connected atoms (insert carbon), removing an atom if it has only two neighbors (cut atom).

We compared our algorithm with SMILES LSTM [39], Graph GA, CReM [36] and EvoMol [27] on GuacaMol [3] benchmark. The final benchmark scores are restricted to the interval [0, 1] (1 is the best score) and are calculated as weighted averages of such molecule scores as: structural features, physicochemical properties, similarity or dissimilarity to other molecules, presence and absence of substructures, functional groups, or atom types. Results of comparison on 10 runs are presented in Table 3.

For the experiment atom types were limited to C, N, O, F, P, S, Cl and Br, while single, double and triple bonds were used. Number of generations set to 3000, population size equals 50 and maximal number of heavy atoms equals 50. To generate only feasible molecular structures the set of possible editing actions is proposed before applying mutations. For initial population the top 100 best scoring molecules are selected from GuacaMol dataset for each benchmark.

According to the Table 3 GOLEM-based solution shows comparable results (within 4% of performance for total score) to the

[2]https://github.com/aimclub/FEDOT
[3]https://github.com/aimclub/BAMT

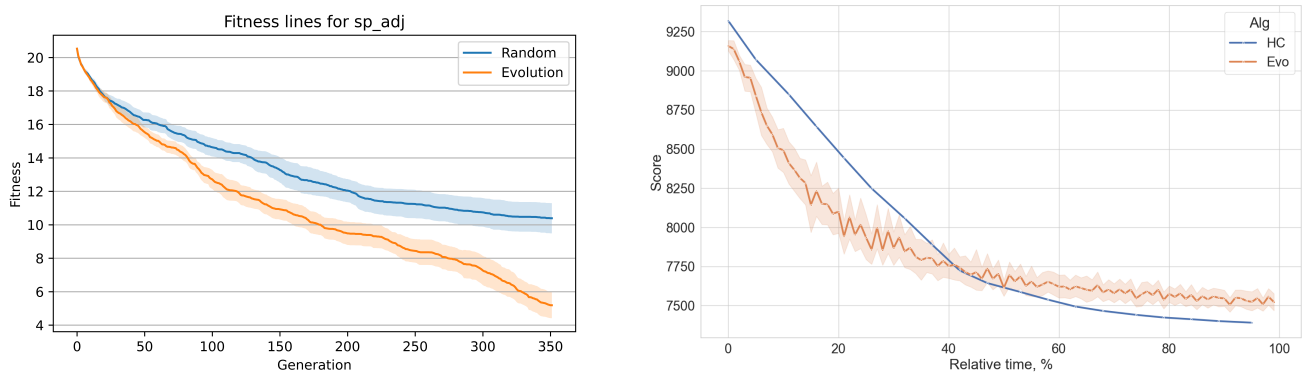

Figure 2: Convergence of a) comparison of random search and evolutionary optimizer for synthetic tree graphs. The convergence of fitness function is averaged for 15 runs; b) BN structure learning from ten runs on Sachs dataset

Table 3: GuacaMol benchmark results. Custom benchmark metric described in text is used for score evaluation.

benchmark	SMILES LSTM	Graph GA	CReM	EvoMol	GOLEM	EvoMol best	GOLEM best
Celecoxib rediscovery	1.000	1.000	1.000	0.978	0.889±0.011	1.000	1.000
Troglitazone rediscovery	1.000	1.000	1.000	1.000	0.926±0.128	1.000	1.000
Thiothixene rediscovery	1.000	1.000	1.000	0.876	0.792±0.026	1.000	0.828
Aripiprazole similarity	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Albuterol similarity	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Mestranol similarity	1.000	1.000	1.000	1.000	0.999	1.000	1.000
C11H24	0.993	0.971	0.966	1.000	0.999	1.000	1.000
C9H10N2O2PF2Cl	0.879	0.982	0.940	0.998	1.000	1.000	1.000
Median molecules 1	0.438	0.406	0.371	0.455	0.382	0.455	0.388
Median molecules 2	0.422	0.432	0.434	0.417	0.405	0.417	0.405
Osimertinib MPO	0.907	0.953	0.995	0.955	0.946	0.969	0.955
Fexofenadine MPO	0.959	0.998	1.000	1.000	0.996	1.000	0.999
Ranolazine MPO	0.855	0.920	0.969	0.966	0.882±0.017	0.957	0.911
Perindopril MPO	0.808	0.792	0.815	0.845	0.751±0.032	0.827	0.782
Amlodipine MPO	0.894	0.894	0.902	0.867	0.918	0.869	0.920
Sitagliptin MPO	0.545	0.891	0.763	0.915	0.739±0.017	0.926	0.765
Zaleplon MPO	0.669	0.754	0.770	0.791	0.746	0.793	0.748
Valsartan SMARTS	0.978	0.990	0.994	0.998	0.988	0.998	0.991
Deco Hop	0.996	1.000	1.000	1.000	0.991	1.000	1.000
Scaffold Hop	0.998	1.000	1.000	1.000	0.984±0.016	1.000	1.000
total	17.341	17.983	17.919	18.061	17.330	18.211	17.693
total MPO	5.637	6.202	6.214	6.339	5.980	6.341	6.080

specialized state-of-the-art algorithms. Given GOLEM's applicability in various graph optimization tasks, these results are indeed promising, underlining its potential in the domain of molecular generation.

5.4 Design of robotic arms

In this case, GOLEM is used to explore the design search space and find the near-optimal terminal graph that describes the linkage mechanism of a robotic arm in rostok[4] library. The graph representation is used to construct a simulation model of grasping, followed by the modeling of the object grasping process under the influence of external disturbances. In order to simulate the rigid body dynamics, the PyChrono [41] physics engine is utilized. The disturbances consist of a smoothly increasing force directed at a 45-degree angle relative to the palm normal. Based on the simulation data, the reward is calculated to evaluate the performance of the grasping

[4]https://github.com/aimclub/rostok

process. The quality assessment is performed using classical grasping metrics, including the fraction of phalanxes contacting with objects, the distance between the object center and the geometric center of contact points, and the ability to withstand external force, which are combined as a weighted sum. A reward value of approximately 9 indicates a successful grasping and holding of the object.

The genetic algorithm itself does not take constraints based on grammar rules into account. Therefore, it could produce graphs that represent non-viable mechanisms. In order to resolve this issue and improve the efficiency of the algorithm, we introduced two modifications: (1) the reward of the non-viable mechanism is set to zero; (2) the initial population is generated using viable designs obtained with graph grammar (GG). The initial population is a formation based on the assumption of diversity. It includes candidates with 1, 2, 3, and 4 fingers in equal quantity; (3) Edge filtering rule for crossover that restricts edges that are not directing to the "body" nodes. The best designs and reward chart are presented in Figure 3. The values of reward function is estimated using simulator.

Furthermore, we employed Monte Carlo Tree Search (MCTS) and graph grammar rules to tackle the same task. The comparison of reward score is shown in the Table 4. The involvement of GOLEM improved the results for 3 of 4 tasks against the MCTS.

Table 4: Comparison of achieved rewards GOLEM evolutionary algorithm and MCTS-based graph grammar rules.

Task	Box	Sphere	Cylinder	Ellipsoid
MCTS + GG	9.082	9.145	7.0	**10.019**
GOLEM	**11.259**	**10.331**	**10.116**	8.919

So, the implemented approach can be considered a hybrid between graph grammar and evolutionary optimization. Also, we extended the abstract implementation of the genetic algorithm obtained from GOLEM with several task-specific evolutionary operators (mutations and crossover) that are specifically engineered to minimize the amount of non-viable mechanisms.

6 CONCLUSION

In the paper, we proposed a flexible and adaptive approach named GOLEM for the automation of design tasks in various scientific fields. It provides the modular interface that takes into account domain-specific objective functions, search space and constraints while using the unified evolutionary core. The experimental validation of the GOLEM is provided for synthetic benchmarks and a set of real-world cases (design of probabilistic models, drug design, design of robotic grippers). GOLEM is available as an open-source tool that can be extended to various applications.

ACKNOWLEDGMENT

This research is financially supported by the Foundation for National Technology Initiative's Projects Support as a part of the roadmap implementation for the development of the high-tech field of Artificial Intelligence for the period up to 2030 (agreement 70-2021-00187)

REFERENCES

[1] Laith Mohammad Abualigah, Mohamed E. Abd Elaziz, Ahmad M. Khasawneh, Mohammad Alshinwan, Rehab Ali Ibrahim, Mohammed Abdulaziz Aide Alqaness, Seyedali Mirjalili, Putra Sumari, and Amir Hossein Gandomi. 2022. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results. *Neural Computing and Applications* 34 (2022), 4081 – 4110. https://api.semanticscholar.org/CorpusID:247412220

[2] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509. https://api.semanticscholar.org/CorpusID:211076047

[3] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. 2019. GuacaMol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling* 59, 3 (2019), 1096–1108.

[4] Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. 2018. GuacaMol: Benchmarking Models for De Novo Molecular Design. *Journal of chemical information and modeling* 59 3 (2018), 1096–1108. https://api.semanticscholar.org/CorpusID:53787096

[5] Irene Córdoba, Eduardo C Garrido-Merchán, Daniel Hernández-Lobato, Concha Bielza, and Pedro Larranaga. 2018. Bayesian optimization of the PC algorithm for learning Gaussian Bayesian networks. In *Advances in Artificial Intelligence: 18th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2018, Granada, Spain, October 23–26, 2018, Proceedings 18*. Springer, 44–54.

[6] Derek G Corneil and Richard M Krueger. 2008. A unified view of graph searching. *SIAM Journal on Discrete Mathematics* 22, 4 (2008), 1259–1276.

[7] Irina Deeva, Anna Bubnova, and Anna V Kalyuzhnaya. 2023. Advanced Approach for Distributions Parameters Learning in Bayesian Networks with Gaussian Mixture Models and Discriminative Models. *Mathematics* 11, 2 (2023), 343.

[8] Qiqi Duan, Guochen Zhou, Chang Shao, Zhuowei Wang, Mingyang Feng, Yijun Yang, Qi Zhao, and Yuhui Shi. 2022. PyPop7: A Pure-Python Library for Population-Based Black-Box Optimization. *ArXiv* abs/2212.05652 (2022). https://api.semanticscholar.org/CorpusID:254564248

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research* 20, 1 (2019), 1997–2017.

[10] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[11] Jonathan Godwin*, Thomas Keck*, Peter Battaglia, Victor Bapst, Thomas Kipf, Yujia Li, Kimberly Stachenfeld, Petar Veličković, and Alvaro Sanchez-Gonzalez. 2020. *Jraph: A library for graph neural networks in jax.* http://github.com/deepmind/jraph

[12] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. 2020. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of The Web Conference 2020*. 1253–1263.

[13] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. 2011. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 9–13.

[14] D Hanzelka. 2008. The use of hybrid genetic algorithms in Bayesian network structure learning from data. *Journal of Applied Mathematics* 1, 2 (2008), 387–396.

[15] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622.

[16] Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. 2020. Deepline: Automl tool for pipelines generation using deep reinforcement learning and hierarchical actions filtering. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2103–2113.

[17] E. Hénault, Maria Harris Rasmussen, and Jan H. Jensen. 2020. Chemical space exploration: how genetic algorithms find the needle in the haystack. *PeerJ Physical Chemistry* (2020). https://api.semanticscholar.org/CorpusID:225459180

[18] Emilie S Henault, Maria H Rasmussen, and Jan H Jensen. 2020. Chemical space exploration: how genetic algorithms find the needle in the haystack. *PeerJ Physical Chemistry* 2 (2020), e11.

[19] Mcdonald Jh. 2014. Handbook of biological statistics.

[20] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, B. Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2020. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics* 13 (2020). https://api.semanticscholar.org/CorpusID:231940292

[21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596, 7873 (2021), 583–589.

[22] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems* 30 (2017).

[23] Amit Dilip Kini, Swaraj Sambhaji Yadav, Aditya Shankar Thakur, Akshar Bajrang Awari, Zimeng Lyu, and Travis Desell. 2023. Co-evolving Recurrent Neural

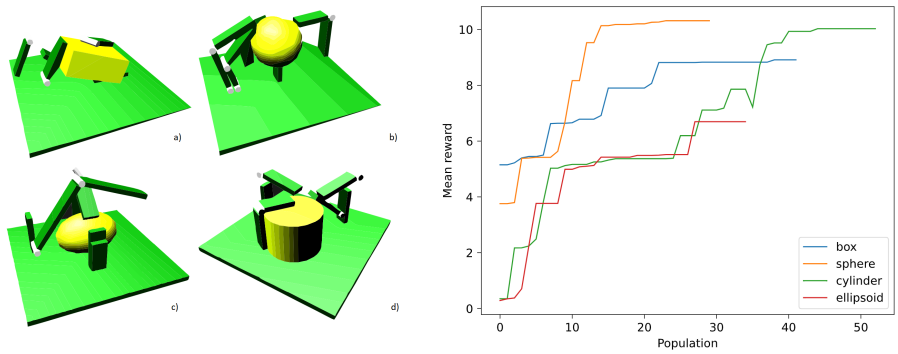

Figure 3: Left. Grippers generated with GOLEM framework: different devices to grasp objects like (a) box, (b) sphere, (c) ellipsoid, (d) cylinder. Right. Mean fitness of 3 best individuals in population show optimization convergence for each task.

Networks and their Hyperparameters with Simplex Hyperparameter Optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 1639–1647.

[24] Neville Kenneth Kitson, Anthony C Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. 2023. A survey of Bayesian Network structure learning. *Artificial Intelligence Review* (2023), 1–94.

[25] Pedro Larranaga, Hossein Karshenas, Concha Bielza, and Roberto Santana. 2013. A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Information Sciences* 233 (2013), 109–125.

[26] Jaehun Lee, Wooyong Chung, and Euntai Kim. 2008. Structure learning of Bayesian networks using dual genetic algorithm. *IEICE transactions on information and systems* 91, 1 (2008), 32–43.

[27] Jules Leguy, Thomas Cauchy, Marta Glavatskikh, Béatrice Duval, and Benoit Da Mota. 2020. EvoMol: a flexible and interpretable evolutionary algorithm for unbiased de novo molecular generation. *Journal of Cheminformatics* 12 (2020). https://api.semanticscholar.org/CorpusID:221716089

[28] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. 2021. GraphDF: A Discrete Flow Model for Molecular Graph Generation. In *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:231749761

[29] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. *ArXiv* abs/2007.08663 (2020). https://api.semanticscholar.org/CorpusID:220633407

[30] Varnavas D Mouchlis, Antreas Afantitis, Angela Serra, Michele Fratello, Anastasios G Papadiamantis, Vassilis Aidinis, Iseult Lynch, Dario Greco, and Georgia Melagraki. 2021. Advances in de novo drug design: from conventional to machine learning methods. *International journal of molecular sciences* 22, 4 (2021), 1676.

[31] Kirill V Nasonov, Dmitriy V Ivolga, Ivan I Borisov, and Sergey A Kolyubin. 2023. Computational Design of Closed-Chain Linkages: Hopping Robot Driven by Morphological Computation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7419–7425.

[32] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. 2020. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4474–4484.

[33] Babatounde Moctard Oloulade, Jianliang Gao, Jiamin Chen, Tengfei Lyu, and Raeed Al-Sabri. 2022. Graph neural architecture search: A survey. *Tsinghua Science and Technology* (2022). https://api.semanticscholar.org/CorpusID:245115768

[34] Damian Owerko, Fernando Gama, and Alejandro Ribeiro. 2019. Optimal Power Flow Using Graph Neural Networks. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), 5930–5934. https://api.semanticscholar.org/CorpusID:204823873

[35] Yun Peng, Byron Choi, and Jianliang Xu. 2021. Graph learning for combinatorial optimization: a survey of state-of-the-art. *Data Science and Engineering* 6, 2 (2021), 119–141.

[36] Pavel Polishchuk. 2020. CReM: chemically reasonable mutations framework for structure generation. *Journal of Cheminformatics* 12, 1 (2020), 1–18.

[37] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv* abs/1703.03864 (2017). https://api.semanticscholar.org/CorpusID:11410889

[38] Marco Scutari. 2017. Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. *Journal of Statistical Software* 77, 2 (2017), 1–20. https://doi.org/10.18637/jss.v077.i02

[39] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. 2018. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science* 4, 1 (2018), 120–131.

[40] Nikita O Starodubcev, Nikolay O Nikitin, Elizaveta A Andronova, Konstantin G Gavaza, Denis O Sidorenko, and Anna V Kalyuzhnaya. 2023. Generative design of physical objects using modular framework. *Engineering Applications of Artificial Intelligence* 119 (2023), 105715.

[41] Alessandro Tasora, Radu Serban, Hammad Mazhar, Arman Pazouki, Daniel Melanz, Jonathan Fleischmann, Michael Taylor, Hiroyuki Sugiyama, and Dan Negrut. 2016. Chrono: An open source multi-physics dynamics engine. In *High Performance Computing in Science and Engineering: Second International Conference, HPCSE 2015, Soláň, Czech Republic, May 25-28, 2015, Revised Selected Papers 2*. Springer, 19–49.

[42] Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. 2022. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*. PMLR, 21658–21676.

[43] Michail Tsagris. 2022. The FEDHC Bayesian network learning algorithm. *Mathematics* 10, 15 (2022), 2604.

[44] W Patrick Walters and Regina Barzilay. 2020. Applications of deep learning in molecule generation and molecular property prediction. *Accounts of chemical research* 54, 2 (2020), 263–270.

[45] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[46] Dennis Wei, Tian Gao, and Yue Yu. 2020. DAGs with No Fears: A closer look at continuous optimization for learning Bayesian networks. *Advances in Neural Information Processing Systems* 33 (2020), 3895–3906.

[47] David Weininger. 1988. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* 28, 1 (1988), 31–36.

[48] Peter Wills and François G. Meyer. 2019. Metrics for graph comparison: A practitioner's guide. *PLoS ONE* 15 (2019). https://api.semanticscholar.org/CorpusID:118711105

[49] Song Xianduo, Wang Xin, Song Yuyuan, Zuo Xianglin, and Wang Ying. 2022. Hierarchical recurrent neural networks for graph generation. *Information Sciences* 589 (2022), 250–264.

[50] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems* 31 (2018).

[51] Dawei Zhou, Lecheng Zheng, Jiejun Xu, and Jingrui He. 2019. Misc-GAN: A multi-scale generative model for graphs. *Frontiers in big Data* 2 (2019), 3.