**LITERATURE REVIEW DOCUMENT WITH REFERENCES**

**Evolutionary Algorithms**

An optimization is a mathematical tool that selects the best solution from available alternatives. Several real-world mathematically defined problems use the optimization concept, such as scheduling, engineering, mathematics, commerce, networks, and economics. Within the last decades, solving optimization problems has caught researchers’ attention. Metaheuristic optimization algorithms (MOA) are commonly utilized to solve those problems[1]. MOA can be classified based on a search strategy (local search and global search), the number of candidate solutions (single solution and population-based), and hybridization (hybrid and memetic). There are several kinds of MOA, such as **Evolutionary Algorithms (EAs)** and swarm intelligence[2-5].

EAs are the most well-known global-search, population based, and memetic MOAs. EAs are heuristic methods inspired by mechanisms that rely on biological evolution, such as reproduction, mutation, and natural selection. In EA, the search space X is a set of chromosomes (i.e., DNA strings) considered candidate solutions for a specific problem. Their fitness is evaluated by objective function (f).

Numerous EAs are developed, such as Genetic Algorithms (GAs)[6-8], Genetic Programming (GP)[9], Evolutionary Programming (EP)[10], Deferential Evolution[11], and Evolution Strategy[12]. The GAs mimic natural selection (i.e., survival of the fittest) and the biological reproduction processes of the fittest individual. The optimal solution (i.e., the fittest individual) is developed from one generation to the next without depending on strict mathematical formulation[13]. Therefore, the optimal solution consists of the best components (i.e., genes) of the fittest individual in previous generations. The simplest form of GA works on a population consisting of individuals (i.e., fixed bit stings).

GA selects a parent pool from the population based on selection criteria to prepare the next generation. The crossover and mutation operators supply the population with new candidates. The crossover operator produces new children (i.e., offspring) by exchanging partial bit strings and inverting bits between two distinct parents. The mutation operator may flip some genes of the new children. GA evaluates each individual using a fitness function. In the last generation, the fittest individual is considered the optimal solution.

**Terminologies and definitions:**

**The fundamental terminologies of GAs are:**

**• Population:** A population is a set of candidate solutions.

**• Chromosome/individual**: A chromosome/individual is a candidate solution. Each chromosome consists of a set of genes and their alleles. A gene is one element position of a chromosome, which is a single bit or short block of adjacent bits [14]. An allele is the gene’s value of a par ticular chromosome

• **Initialization**: Initialization is the first process in GA responsible for preparing the initial population. The GA fills the population with random candidate solutions (i.e., individuals).

• **Evaluation**: An evaluation process is responsible for determining the fitness level of an individual. GA utilizes a problem-dependent fitness function. This operation is triggered once a new individual is produced.

• **Selection**: A selection process is essential in GA to select the parents for the crossover operation. The simplest selection technique is based on the fitness value, where the better solutions have the highest probability of being selected than the worse ones.

• **Crossover**: A crossover operation is a recombination process responsible for producing new offspring.

• **Mutation**: A mutation operation is a random deformation of the individual with a specific probability.

• **Replacement**: A replacement operation is responsible for preparing the population for the next generation. The basic technique selects the fittest individuals of the cur rent generation (i.e., parents and new offspring) to pre pare the next generation.

* **Stop criteria**: Stop criteria are specified to determine when to stop the GA and select the optimal solution.

Typically, at least one of the following criteria is specified:

* Reach the maximum number of generations.
* Find an individual in the population with a fitness value lower/higher than a threshold.

**Genetic algorithm basics and operations:**

A GA starts with initializing a population of size N. The fitness value of each individual in the population is evaluated using a f itness function. Then, the process enters a loop for a specified number of generations (maxGan) where GA uses the current generation (currentGen) to generate the next generation. A set of individuals is selected to form the parents pool. The parent pool helps in producing new offspring using crossover and muta tion techniques. The search process is terminated when a stop criterion is satisfied (i.e., reaching the maxGan generation or f inding an individual who satisfied a stop criterion).

**Population diversity**. A population with a low diversity level leads to a GA like a local search algorithm with an additional overhead from maintaining many similar solutions [15]. Premature convergence refers to a popula tion containing similar individuals before exploring the search space. A diverse population helps the GA explore different regions of the search space, thus reducing the probability of being stuck in the local optimum of a bad f itness degree.

**Population size**. The population size is fixed; thus, it significantly impacts GA performance. The probability of covering promising regions of the search space decreases as the search space dimensionality increases[16]. Thus, selecting a small population size reduces population diversity quickly after applying the crossover operation. On the other hand, selecting a large population waste computational resources.

The representation of the individual in the population depends mainly on the problem. An individual represents by using a bit string (i.e., simplest and most popular encoding) or non-binary representation. Generally, the individual con sists of a set of genes, and each gene has an allele.

**Fitness function:** The fitness function is an essential component of any GA used to measure the fitness value of individuals. A fitness function depends on a single objective function or multi objective function. The objective function is a function that measures the performance concerning a set of parameters (i.e., alleles of the individual). In contrast, the fitness function measures the reproduction probability of each individual depending on the objective function[17].

**Multi-Objective Optimization**

Multi-objective optimization involves optimizing a number of objectives simultaneously. The problem becomes challenging when the objectives are of conflict to each other, that is, the optimal solution of an objective function is different from that of the other. In solving such problems, with or without the presence of constraints, these problems give rise to a set of trade-off optimal solutions, popularly known as Pareto-optimal solutions. All solutions in a Pareto optimal set are characterised by the fact that there are no other individual solutions that have a higher(or equal) fitness in all objective functions. Together, the set of Pareto optimal solutions form an optimal envelope in objective space known as the Pareto front. The Pareto front provides a family of solutions, all equivalent in principle, aiding domain experts to make choices when trade-offs between objectives are known beforehand [34]. Due to the multiplicity in solutions, these problems were proposed to be solved suitably using evolutionary algorithms which use a population approach in its search procedure.[18]. A multi-objective optimization problem involves a number of objective functions which are to be either minimized or maximized. As in a single-objective optimization problem, the multi-objective optimization problem may contain a number of constraints which any feasible solution (including all optimal solutions) must satisfy.

**NSGA-II**

NSGA-II (Non-dominated Sorting Genetic Algorithm II)[19] is a powerful evolutionary algorithm designed for multi-objective optimization problems, efficiently discovering a diverse set of Pareto-optimal solutions. The algorithm operates by maintaining a set of candidate solutions (or individuals) and evolves these through processes of selection, crossover, mutation, and replacement.

**Key Components of NSGA-II**

1. **Population Initialization**: The NSGA-II begins with a randomly initialized population P0​ of size N across the search space.
2. **Non-Dominated Sorting**: One of the key innovations of NSGA-II is its fast non-dominated sorting algorithm, which classifies the individuals into different fronts based on Pareto dominance.

* A solution x dominates another solution y if: fi​(x)≤fi​(y)∀i and ∃j:fj​(x)<fj​(y)
* The population is categorized into multiple fronts:
* **Front 1**: Comprises the best solutions that are non-dominated.
* **Front 2**: Contains solutions dominated only by those in Front 1, and so forth.

This sorting allows the algorithm to focus on the best solutions at each generation while identifying the Pareto front.

1. **Crowding Distance Calculation**: NSGA-II incorporates a crowding distance mechanism to maintain diversity among solutions within the same non-dominated front. The crowding distance d(x) for a solution x is defined as: d(x)=∑k=1m​dk​(x) where dk​(x) is the crowding distance with respect to each objective function. For an individual in a front, it is calculated by:

* Normalizing the objective values of neighboring solutions.
* Assigning higher crowding distance values to solutions in less crowded regions.

This allows NSGA-II to retain individuals that are spaced far apart in the objective space, thereby enhancing population diversity.

1. **Selection Process**: The selection of parents for the next generation is influenced by both the non-domination rank and the crowding distance. NSGA-II employs a binary tournament selection process:

* Two individuals are randomly selected.
* The individual with the better rank (lower rank number) is favored.
* In cases where the ranks are equal, the individual with the higher crowding distance is selected.

1. **Genetic Operators (Crossover and Mutation)**: Standard genetic operators are applied for generating the offspring population Qt​:

* **Crossover** can utilize methods such as simulated binary crossover (SBX).
* **Mutation** often involves a polynomial mutation operator to introduce variations.

1. **Population Replacement**: To form the new generation, the current population Pt​ and the offspring population Qt​ are combined. The next population Pt+1​ is selected based on the sorted fronts and crowding distances. The process involves:

* Sorting the combined population by non-domination ranks and selecting the best N individuals for the next generation.

In contrast to NSGA-II, the NSGA-III algorithm[37,38] uses reference directions instead of a crowding distance to enforce diversity in the selection of solutions within the splitting front. Reference directions are determined by a predetermined set of points on the unit simplex in fitness space. Each reference direction is defined as a ray originating from the origin and passing through exactly one of these points. NSGA-III assigns a reference directon to each solution in the population based on the nearest perpendicular distance(in normalised fitness space) to the corresponding direction. In the splitting front selection procedure, the NSGA-III algorithm prioritises refer ence directions that are underrepresented in the current surviving evolutionary population. If a reference direction does not have any solution assigned to it after reaching the splitting front, then the molecule in the splitting front with the smallest perpendicular distance to this direction is selected for survival. If all underrepresented refer ence directions have been assigned one surviving solution, and the maximum size of the surviving population has not been reached, the remaining solutions are selected by a stochastic procedure. Note that NSGA-III selects the solutions in the fronts before the splitting front in its entirety, like in NSGA-II. However, contrary to NSGA-II's crowding distance which is calculated within the splitting front, the reference directions used in NSGA-III take into account the diversity of the entire surviving population [34]. However, later analyses [39,40] have shown that for a wide range of computational experiments, NSGA-III does not consistently outperform NSGA-II in every use case.

**Pymoo**

Pymoo is a multi-objective optimization framework in Python designed to provide comprehensive tools for multi-objective optimization tasks [20]. It offers customizable implementations, allowing modification and extension of algorithms with custom operators, and includes single, multi-, and many-objective test problems with automatic differentiation for gradients. The architecture consists of optimization problems, algorithms, and analytics, each with sub-modules. Problems are categorized into single, multi, and many-objective test problems with available gradients and parallelization techniques. The optimization module provides sub-modules for algorithms, evolutionary operators, termination criteria, and decomposition methods. Available algorithms include NSGA-II, NSGA-III, MOEAD, and others, each customizable with different parameters. Evolutionary operators include sampling (random, Latin-Hypercube), crossover (one/two-point, uniform, half uniform, SBX), and mutation (polynomial, bitflip).

Here is a simple code snippet illustrating how to define a multi-objective problem and run an optimization using pymoo [20]:

import pymoo

from pymoo.optimize import minimize

from pymoo.factory import get\_problem, get\_algorithm

# Define a multi-objective problem

problem = get\_problem("zdt1") # Example: ZDT1 problem

# Create an algorithm instance algorithm = get\_algorithm("nsga2", pop\_size=100)

# Run the optimization

res = minimize(problem,

algorithm,

termination=('n\_gen', 100),

seed=1)

**De Novo Molecular Design and Generative Models**

The field of drug discovery aims to invent useful new medicines for many reasons. New medicine can facilitate more effective and afffordable treatment of known diseases, make previously incurable or unknown conditions curable, and improve health in general. A major drive behind said research is that a plethora of undiscovered drugs exist that have the potential to aid in the creation of such novel medications. De novo molecular design is the process of automatically proposing novel chemical structures that optimally satisfy a desired molecular profile [21]. Traditionally, visual screening(VS) is undertaken to identify molecules likely to exhibit desirable experimental outcomes. A key difference, compared with *de novo* design, is the source of the molecules considered: where structures are known a priori in VS, in *de novo* design we seek to generate the structures to be evaluated. De novo design approaches require three components : (1) Molecule generation, (2) A way to score the molecules, and (3), a way to optimize or search for better molecules with respect to the scoring function [43]. De novo design has a rich history in chemoinformatics and has received recent attention as ML methodologies continue to open new possibilities for navigating and sampling large search spaces without reverting to explicit rules or expert knowledge. However, training-free optimisation algorithms that comprehensively traverse and explore chemical space have been shown to be more efficient [34]. The search space is known as the chemical space and is so unimaginably vast that it is practically impossible to explore exhaustively [22]. Instead, a more narrow and targeted search is required. Researchers now routinely screen millions of compounds in the search for some that are biologically active. Yet even the compound files of the largest pharmaceutical companies(which typically contain approximately 10^6 compounds) offer only a cursory examination of all the possible organic componds that comprise ‘chemical space’. Also, not all biologically active compounds have the desired physiochemical properties to be a drug. A biologically active compound may be lipophilic (greasy) to be orally absorbed, too polar to cross the gastrointestinal wall or may have too much vulnerable chemical functionality that can be attacked by metabolizing systems in the liver, and therefore not remain intact for long enough to have a useful *in vivo* effect. [23]. De novo design methods are often evaluated by their performance on standalone toy tasks, such as maximizing the quantitative estimate of drug-likeness(QED) [24]. But in other fields of machine learning, for example, computer vision and natural langiage processing, standardized benchmarks have triggered rapid progress [41,42]. Similarly, the field of de novo molecular design can benefit from the introduction of standardized benchmarks. They allow for a straightforward survey and comparison of existing models and provide insight into the strengths and weaknesses of models, which is valuable information to improve current approaches. Another more suitable and real-world oriented method is using the Molecular Sets(MOSES) benchmark which includes a set of distribution learning tasks along with measures of molecule validity and uniqueness [25]. The aim of distribution learning tasks is to measure the structural diversity and relevance of proposed compounds by comparing the generated chemical space to known chemical structures; MOSES also considers scaffold and fragment diversity. A more suitable alternative is the GuacaMol framework, which defines a suite of benchmarks and implement it as a Python package designed to allow researchers to assess their models easily. The GuacaMol benchmarks suite incudes, in addition to distribution benchmarks, a more applied set of goal-directed tasks, which imitate discrete uses of de novo design tools [26].

**Molecular representation**

Computational methods for evaluating chemical structures must rely upon a suitable molecular representation, that is, the form in which a molecular structure is seen by a subsequent algorithm. Molecular representation is a broad topic [27]; for example, methods can encode the presence or absence of functional groups, express a molecule as its topological graph, or include 3D information describing bond angles. Among de novo design methods, common molecular representations are text based, such as the simplified molecular input line entry system(SMILES) [28], and graph based where the molecular generator might operate explicitly on the molecular topology. However, a significant issue with SMILES is the low probability of random strings forming valid compound structures. There is no guarantee that a new string representation, created from the combination of parts of SMILES representations, will correspond to a feasible compound. This problem implies that the string representation of offspring generated through EA crossover does not always correspond to a viable structure. Consequently, the use of SMILES can result in inefficient exploration [29]. On the other hand, SELF-referencing Embedded Strings(SELFIES) [30], a recent method for converting to string representation, guarantees that random strings will form valid structures. Therefore, SELFIES is expected to be effective in EA-based exploration. SELFIES ensures that every combination of symbols maps to a chemically valid graph, thereby preventing the generation of invalid molecules. SELFIES employs a formal grammar-based method where its derivation rules ensure that every combination of symbols corresponds to a chemically valid graph. This attribute effectively prevents the production of invalid molecules, facilitating more efficient compound identification in evolutionary com putation. While SELFIES is suitable for representing typical organic molecules, it does not encompass all molecular types.

**Challenges designing an objective function**

An outstanding challenge for de novo design is for desired property profiles to reflect the needs of medicinal chemistry more accurately. Although it is useful to demonstrate that methods can optimize molecules toward calculated molecular property profiles, similarity measures or quantitative structure–activity relationship (QSAR) models, drug discovery is multifaceted and current de novo design efforts are limited by a narrow view of the overall process. Although there is an ongoing need to improve predictive models of complex biological responses, multi-objective optimization (MOO) aims to coalesce signals from several weak scorers using data fusion concepts, such as Pareto optimality [31]. Often there are additional requirements that make it necessary to optimise for additional properties such as low toxicity[35], high synthesizability [36], or off-target activity, in which case MOO is necessary, and a trade-off between different(and possibly competing optimisation objectives has to be defined. The design of effective MOO profiles is nontrivial and often makes use of normalization functions and scaling protocols when combining multiple objectives [32]. It is usually necessary to experiment with several iterations between scoring function refinement and molecular generation.

**SELFIES**

A significant fraction of the resulting SMILES strings do not correspond to valid molecules. They are either syntactically invalid, i.e, do not even correspond to a molecular graph, or they violate basic chemical rules, such as the maximum number of valence bonds between atoms. Researchers have proposed many special-case solutions for overcoming these problems. For example, by adapting the machine learning models such that they deal with invalidity [33]. While this solves the problems for specific models, it does not provide a universal solution for all current(and future) possible models. An alternative way is SELFIES, string-based representation of molecular graphs that is 100 % robust. Each SELFIES corresponds to a valid molecule, even entirely random strings. Furthermore, every molecule can be described as a SELFIES. SELFIES can be used as a direct input into current and even future generative models, without the requirement to adapt the model. The unique factor of SELFIES to ensure valdity of generated molecules is that it is based on context-free grammar that includes built-in error correction. Therefore, even if you mutate a SELFIES string randomly, the result will still be a valid molecule.

For example, the SELFIES representation of ethanol(CCO) would look like [C][C][O] but it often includes extra tokens to encode bonding and ensure valence constraints are met. A more realistic SELFIES string for ethanol might be – [C][C][O][Branch1][Ring1].

**Assessing De Novo Design Techniques with GuacaMol**

To profile models for de novo molecular design, the framework differentiate between their two main use cases:

• Given a training set of molecules, a model generates new molecules following the same chemical distribution.

• A model generates the best possible molecules to satisfy a predefined goal.

The collection of benchmarks assess both facets defined here. They are referred to as distribution-learning benchmarks and goal-directed benchmarks, respectively.

The two benchmarks categories are evaluated independently to afford models as much flexibility as possible without penalty, since there is no one-to-one correspondence between distribution-learning and goal-directed tasks.

**1. Distribution Learning Benchmarks:** Models for de novo drug design often learn to reproduce the distribution of a training set or use this training set to derive molecular fragments before generating targeted molecules. This allows some model architectures to learn the syntax of molecules in the selected molecular representation and often accelerates the goal-directed tasks. The distribution-learning benchmarks assess how well models learn to generate molecules similar to a training set, which in this work is a standardized subset of the ChEMBL database [44].

Five benchmarks are considered for distribution learning:

Validity, Uniqueness,Novelty, Fréchet ChemNet Distance, KL Divergence

**2.** **Goal-Directed Benchmarks:** The molecule score reflects how well a molecule fulfills the required property profile(sometimes also called multiproperty objective/MPO). The goal is to find molecules that maximise the scoring function. Concretely, the models are asked to generate a given number of molecules with high scores for a given function. The models have access to the scoring function and can iteratively improve their best molecule guesses, without knowing explicitly what the scoring function calculates. In general, the function to optimize will be defined as the combination of one or several functions, representing different molecular features such as

• Structural features. Examples: molecular weight, number of aromatic rings, number of rotatable bonds.

• Physicochemical properties. Examples: TPSA, logP.

• Similarity or dissimilarity to other molecules.

• Presence and absence of substructures, functional groups, or atom types.

The majority of benchmarks define complex combinations of such features. In addition, some benchmarks fall into special categories:

Similarity, Rediscovery, Isomers, Median Molecules

**Compound Quality :** In previous works, several authors have highlighted that unrestricted de novo design algorithms can generate compounds that are potentially unstable, reactive, laborious to synthesize, or simply unpleasant to the eye of medicinal chemists [45,46,47]. Systems proposing too many unsuitable compounds lose trust and will not be accepted by medicinal chemists. Unfortunately, explicitly encoding all the background knowledge medicinal chemists acquire with experience as an exhaustive list of unstable or undesirable substructures is challenging, if not impossible, due to the inherent subjectivity and context dependency – for example, a toxicity risk might be assessed differently in oncology and diabetology. GuacaMol employs Walter’s rd\_filters implementation [48] using the SureChembl, Glaxo, PAINS(all retrieved from ChEMBL), and in-house rule collections, to calculate the compound quality metrics.

GuacaMol also encompasses the QED benchmark which aim to optimize the “quantitative estimate of druglikeness”(QED) [49], an empirical measure of drug-likeness, similar to the Lipinski’s rule of 5 [50]. Even though optimization of drug-likeness alone is not a particularly useful objective in drug discovery, it has been used in several publications as a target for de novo molecular design.

**Possible Alternative Evolutionary Algorithms**

There are alternative EAs to NSGA-II and NSGA-III which are specifically tailored for de novo drug design. Among these, EvoMol [51] and MolFinder [52] comes across as promising . Although NSGA‐II has been widely used for Pareto‐based optimization in drug design, both EvoMol and MolFinder introduce domain‐specific operations and representations that can potentially improve the efficiency, diversity, and quality of the generated molecules.

**EvoMol**

EvoMol is a de novo design algorithm that employs an evolutionary approach to generate novel molecules. Its design centers on iterative mutation and crossover operations applied directly to molecular representations—often using SMILES strings or, in more recent adaptations, SELFIES. Key advantages of EvoMol include:

* **Robust Chemical Validity**: By leveraging either the SMILES or the more robust SELFIES representations, EvoMol can ensure a high validity rate among generated compounds. In particular, the SELFIES variant avoids many common syntactical pitfalls that plague SMILES‐based generators.
* **Efficient Exploration of Chemical Space**: EvoMol uses a diverse set of mutation operators that introduce substitutions, insertions, and deletions, allowing the algorithm to explore both local and global regions of chemical space. This flexible mutational scheme helps prevent premature convergence, a common challenge in traditional EAs.
* **Performance on Benchmark Tasks**: When evaluated using benchmarks such as GuacaMol, EvoMol has demonstrated competitive performance in metrics including novelty, uniqueness, and the ability to rediscover known bioactive molecules. In some instances, it outperforms NSGA‐II on rediscovery and novelty scores, likely due to its tailored mutation strategy that better balances exploitation (refinement of high-scoring leads) and exploration (diversity generation).

**MolFinder**

MolFinder represents another approach that combines chemical heuristics with evolutionary operations. Unlike EvoMol, which is typically described in the context of string-based representations, MolFinder operates on molecular graphs. Its distinctive features include:

* **Graph-based Optimization**: MolFinder performs genetic operations directly on molecular graphs. By doing so, it more naturally respects the inherent topology of molecules, which can lead to more chemically intuitive crossovers and mutations.
* **Enhanced Diversity and Avoidance of Convergence Issues**: MolFinder incorporates advanced diversity-preserving strategies such as controlled mutation rates and specific recombination operators designed to maintain chemical diversity. This helps the algorithm avoid common pitfalls like local minima, and it has been shown to produce libraries with high internal diversity.
* **Benchmark Performance**: In comparative studies on the GuacaMol benchmarks, MolFinder has shown strong performance in metrics such as internal diversity and the generation of novel scaffolds. Its graph-based operations often lead to a higher proportion of structurally diverse compounds compared to some string-based methods and can match or exceed NSGA‐II in multi-objective optimization tasks that emphasize chemical diversity and synthetic accessibility.

**Comparative Advantages and Performance**

Both EvoMol and MolFinder offer significant advantages over standard multi-objective EAs like NSGA‐II when applied to de novo molecular design:

* **Chemical Representation**: While NSGA‐II relies on generic representations and requires careful constraint handling to ensure chemical validity, EvoMol and MolFinder use domain-adapted representations (SMILES/SELFIES for EvoMol and molecular graphs for MolFinder) that inherently capture chemical rules and enhance validity.
* **Exploration vs. Exploitation**: NSGA‐II excels at finding well-balanced Pareto fronts but may sometimes be limited by its fixed selection and crossover schemes. In contrast, EvoMol’s flexible mutational strategies and MolFinder’s graph-based operators allow for both fine-tuned local optimizations (exploitation) and broader chemical space exploration.
* **Benchmark Metrics (GuacaMol)**: Studies comparing these approaches on standardized tasks (e.g., rediscovery, novelty, uniqueness, property optimization) suggest that EvoMol can sometimes achieve higher novelty and rediscovery scores, whereas MolFinder often demonstrates superior internal diversity. Both methods have been reported to match or even outperform NSGA‐II in certain tasks, particularly when drug-likeness (as measured by metrics like QED and synthetic accessibility scores) is taken into account.
* **Computational Efficiency**: By incorporating chemical knowledge directly into their mutation and recombination operations, EvoMol and MolFinder tend to reduce the number of invalid or chemically nonsensical molecules generated during evolution. This not only improves the quality of the candidate molecules but also enhances the overall computational efficiency relative to approaches that rely solely on post-hoc filtering (a common burden in NSGA‐II implementations).

**DeLA-DrugSelf**

DeLA-DrugSelf [54] is a novel deep learning-based framework designed for multi-objective de novo drug design, leveraging the SELFIES molecular representation to ensure robust chemical validity during molecule generation. It is an enhanced successor to the earlier DeLA-Drug model, which was limited to SMILES-based single-character substitutions. DeLA-DrugSelf extends this concept significantly by incorporating a wider range of mutational operations—including insertions, deletions, and substitutions—directly on SELFIES strings, which enables it to perform more expressive molecular modifications suitable for scaffold decoration and lead optimization. This flexibility allows the model to explore the chemical space more effectively without relying on hand-crafted crossover or repair mechanisms typically required in SMILES-based genetic algorithms.

Unlike methods such as NSGA-II, NSGA-III, or MOEA/D, which are classical multi-objective evolutionary algorithms that rely on explicit population-based operations (e.g., crossover, selection, non-dominated sorting), DeLA-DrugSelf operates within a mutation-driven generative loop guided by a Pareto-based scoring mechanism. Here, each generated molecule is evaluated across multiple objectives, and the algorithm retains those that belong to the non-dominated Pareto front. This enables simultaneous optimization of several molecular properties, similar in spirit to what evolutionary algorithms accomplish, but using neural network-guided generation rather than population evolution.

A core innovation of DeLA-DrugSelf lies in its handling of a critical issue with SELFIES-based generation: SELFIES "collapse", where mutated strings can map to excessively small or degenerate molecular graphs. The algorithm incorporates explicit checks to filter out such collapsed structures, ensuring that only meaningful and diverse molecular candidates proceed through the optimization pipeline. This feature significantly boosts the reliability of the generated compounds.

DeLA-DrugSelf has been successfully applied to the optimization of ligands targeting cannabinoid receptor 2 (CB2R), showcasing its ability to produce bioactive molecules with favorable multi-objective profiles. The optimization process was benchmarked using datasets and scoring functions similar to those used in platforms like GuacaMol and ZINC, demonstrating the model’s applicability in real-world drug discovery scenarios. Furthermore, it is implemented as a user-accessible web platform, allowing researchers to interactively generate and evaluate candidate molecules based on custom multi-objective criteria.

In summary, DeLA-DrugSelf provides a data-driven, mutation-based alternative to classical MOEAs in de novo molecular design. While it achieves similar goals—optimizing for drug-likeness, synthesizability, and bioactivity—it does so through deep learning and Pareto-based mutation rather than crossover-based evolution. Its compatibility with SELFIES and its ability to handle complex optimization objectives make it a powerful addition to modern CADD toolkits, particularly in settings where interpretability, robustness, and chemical validity are critical.

**Other Related Works**

Jensen [55] implemented a graph-based Genetic Algorithm with Monte Carlo Tree Search. It matches machine learning based techniques in compound quality and exceeds in computing speed. Based on his findings, Jensen makes a strong argument in favour of EAs regarding their simplicity, and in a follow up work, Henault, Rasmussen, and Jensen [56] highlight this argument by emphasizing the ability of Eas for effective chemical space exploration.

**Optimized Drug Design using Multi-Objective Evolutionary Algorithms with SELFIES and NSGA-II**

This study presents a comprehensive pipeline for de novo drug design using MOEAs in combination with SELFIES. The process begins with data acquisition from the ZINC database [53], from which compounds were filtered based on Lipinski’s Rule of Five to ensure drug-likeness, narrowing the dataset to molecules with molecular weights between 250–350 Da and a LogP below 5. From over 400 million compounds, a random 1% sample (~4 million) was drawn and further refined using MOSES benchmark filters, resulting in a curated subset of 3.5 million compounds. These were initially represented in SMILES format and then converted to SELFIES to ensure all genetic operations during evolution would yield chemically valid molecules without requiring post-processing or repair.

To optimize for both drug-likeness and synthesizability, the study employed three well-known MOEAs—NSGA-II, NSGA-III, and MOEA/D—using a set of multi-objective criteria: QED (quantitative estimate of drug-likeness), SA score (synthetic accessibility), and five goal-directed tasks from the GuacaMol benchmark suite (Cobimetinib, Fenofexadine, Osimertinib, Pioglitazone, and Ranolazine). Genetic operations included one-point crossover and targeted SELFIES-aware mutation (insertion, deletion, substitution), with special strategies implemented in MOEA/D to prevent premature convergence, such as selective offspring replacement based on similarity thresholds.

The experiments involved running each algorithm across 5 tasks, two population sizes (100 and 500), and 10 repetitions per configuration, resulting in a total of 100 experimental runs. Evaluations focused on convergence (using the running metric), diversity (via extended Baroni-Urbani-Buser similarity indices), and the quality and novelty of the resulting compounds. Results indicated that NSGA-II consistently produced the largest and most diverse Pareto-optimal sets across all scenarios, while NSGA-III showed superior performance in terms of solution quality for larger populations. MOEA/D performed competitively in smaller population sizes but was generally outperformed by NSGA-III as the population increased.

Crucially, the analysis revealed that a substantial number of the evolved compounds were not present in the original ZINC database, suggesting the potential discovery of novel, synthesizable drug-like molecules. These compounds were further assessed using the SwissADME tool, which confirmed that many of them fit within the desirable chemical space and passed multiple drug-likeness filters, including PAINS and Brenk alerts. The use of SELFIES proved vital in enabling a smooth and effective evolutionary process, avoiding invalid solutions and promoting efficient exploration of the chemical space. Overall, the study demonstrates the feasibility and effectiveness of using MOEAs with SELFIES for drug discovery, offering a simpler, interpretable, and computationally efficient alternative to deep learning approaches, and paving the way for further automation and refinement in CADD pipelines.

**REFERENCES**

1. BoussaïD, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences, 237*, 82–117.

2. Agushaka, J. O., Ezugwu, A. E., & Abualigah, L. (2022). Dwarf mongoose optimization algorithm. *Computational Methods in Applied Mechanics and Engineering, 391*, 114570.

3. Ezugwu, A. E., Agushaka, J. O., Abualigah, L., Mirjalili, S., & Gandomi, A. H. (2022). Prairie dog optimization algorithm. *Neural Computing and Applications, 34(22)*, 20017–20065.

4. Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z. W., & Gandomi, A. H. (2022). Reptile search algorithm (RSA): A nature-inspired metaheuristic optimizer. *Expert Systems with Applications, 191*, 116158.

5. Oyelade, O. N., Ezugwu, A. E-S., Mohamed, T. I., & Abualigah, L. (2022). Ebola optimization search algorithm: A new nature-inspired metaheuristic optimization algorithm. *IEEE Access, 10*, 16150–16177.

6. Holland, J., & Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Massachusetts.

7. Mattfeld, D. C. (2013). *Evolutionary search and the job shop: Investigations on genetic algorithms for production scheduling.*

8. Arabali, A., Ghofrani, M., Etezadi-Amoli, M., Fadali, M. S., & Baghzouz, Y. (2013). Genetic-algorithm-based optimization approach for energy management. *IEEE Transactions on Power Delivery, 28(1)*, 162–170.

9. Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing, 4(2)*, 87–112.

10. Kim, J.-H., & Myung, H. (1997). Evolutionary programming techniques for constrained optimization problems. *IEEE Transactions on Evolutionary Computation, 1(2)*, 129–140.

11. Storn, R., & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11(4)*, 341–359.

12. Rechenberg, I. (1973). *Evolution strategy: Optimization of technical systems by means of biological evolution.* Fromman Holzboog Stuttgart, 104, 15–16.

13. Man, K.-F., Tang, K.-S., & Kwong, S. (1996). Genetic algorithms: Concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics, 43(5)*, 519–534.

14. Mitchell, M. (1998). *An introduction to genetic algorithms.*

15. Sudholt, D. (2018). The benefits of population diversity in evolutionary algorithms: A survey of rigorous runtime analyses. *arXiv preprint arXiv:1801.10087.*

16. Kazimipour, B., Li, X., & Qin, A. K. (2014). A review of population initialization techniques for evolutionary algorithms. *2014 IEEE Congress on Evolutionary Computation (CEC),* IEEE, pp. 2585–2592.

17. Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing, 4(2)*, 65–85.

18. Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction - Kalyanmoy Deb

19. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans EvolComput 2002;6(2):182–97

20. Pymoo: Multi-Objective Optimization in Python JULIAN BLANK AND KALYANMOY DEB , (Fellow, IEEE)

21. De novo molecular design and generative models – Joshua Meyers, Benedek Fabian, and Nathan Brown

22. Pavel Polishchuk. “CReM: chemically reasonable mutations framework for structure generation”. In: Journal of Cheminformatics 12.1 (2020), pp. 1–18.

23. Navigating chemical space for biology and medicine – Christopher Lipinski, Andrew Hopkins

23. G.R. Bickerton, G.V. Paolini, J. Besnard, S. Muresan, A.L. Hopkins, Quantifying the chemical beauty of drugs, Nat Chem 4 (2012) 90–98.

24. Polykovskiy D, Zhebrak A, Sanchez-Lengeling B, Golovanov S, Tatanov O, Belyaev S, et al. Molecular Sets (MOSES): a benchmarking platform for molecular generation models. arXiv [csLG]. Published online November 29, 2018. http:// arxiv.org/abs/1811.12823.

25. N. Brown, M. Fiscato, M.H.S. Segler, A.C. Vaucher, GuacaMol: Benchmarking models for de novo molecular design, J Chem Inf Model 59 (2019) 1096–1108.

26. B. Sanchez-Lengeling, A. Aspuru-Guzik, Inverse molecular design using machine learning: Generative models for matter engineering, Science 361 (2018) 360–365.

27. David Weininger. “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules”. In: Journal of chemical information and computer sciences 28.1 (1988)

28. Optimized Drug Design using Multi-Objective Evolutionary Algorithms with SELFIES

29. Mario Krenn et al. “Self-referencing embedded strings (SELFIES): A 100Science and Technology 1.4 (2020), p. 045024.

30. C.A. Nicolaou, N. Brown, Multi-objective optimization methods in drug design, Drug Discov Today Technol 10 (2013) e427–e435

31. C. Grebner, H. Matter, A.T. Plowright, G. Hessler, Automated de novo design in medicinal chemistry: which types of chemistry does a generative neural network learn?, J Med Chem 63 (2020) 8809–8823

32. Tengfei M, Chen J and Xiao C 2018 Constrained generation of semantically valid graphs via regularizing variational autoencoders Advances in Neural Information Processing Systems 31 (NIPS 2018) pp 7113–24

33. Liu Q, Allamanis M, Brockschmidt M and Gaunt A 2018 Constrained graph variational autoencoders for molecule design Advances in Neural Information Processing Systems 31 (NIPS 2018) pp 7795–804

34. Graph-based molecular Pareto optimisation – Jonas Verhellen

35. D. A. Smith, A. Harrison and P. Morgan, Chem. Res. Toxicol., 2011, 24, 463–474

36. W. Gao and C. W. Coley, The Synthesizability of Molecules Proposed by Generative Models, 2020.

37. K. DebandH.Jain, IEEE Trans. Evol. Comput., 2014, 18, 577 601.

38. H.Jain and K. Deb, IEEE Trans. Evol. Comput., 2014, 18, 602 622

39. H. Ishibuchi, R. Imada, Y. Setoguchi and Y. Nojima, 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 3045–3052.

40. G. Campos Ciro, F. Dugardin, F. Yalaoui and R. Kelly, IFAC PapersOnLine, 2016, 49, 1272–1277.

41. Deng, J.;Dong,W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. Imagenet:Alarge-scalehierarchicalimagedatabase.IEEEConferenceon ComputerVisionandPatternRecognition;IEEE,2009;pp248−255.

42. Mayr, A.; Klambauer, G.; Unterthiner, T.; Hochreiter, S. DeepTox: toxicitypredictionusingdeeplearning.Front.Environ.Sci. 2016,3,80.

43. Hartenfeller,M.;Schneider,G.Enablingfuturedrugdiscoveryby denovodesign.WileyInterdiscip.Rev.:Comput.Mol.Sci.2011,1,742− 759.

44. Mendez, D.; Gaulton, A.; Bento, A. P.; Chambers, J.; de Veij, M.; Felix, E.; Magarinos, M. P.; Mosquera, J. F.; Mutowo, P.; Nowotka, M.; Gordillo-Maranon, M.; Hunter, F.; Junco, L.; Mugumbate, G.; Rodriguez-Lopez, M.; Atkinson, F.; Bosc, N.; Radoux, C. J.; Segura Cabrera, A.; Hersey, A.; Leach, A. R. ChEMBL: towards direct deposition of bioassay data. Nucleic Acids Res. 2019, 47, D930.

45. Ertl,P.;Lewis,R.IADE:asystemforintelligentautomaticdesign of bioisostericanalogs. J.Comput.-AidedMol.Des. 2012, 26, 1207− 1215.

46. Hartenfeller,M.;Schneider,G.Enablingfuturedrugdiscoveryby denovodesign.WileyInterdiscip.Rev.:Comput.Mol.Sci.2011,1,742− 759.

47. Gasteiger, J. De novo design and synthetic accessibility. J. Comput.-Aided Mol. Des. 2007, 21, 307−309.

48. Walters, P. rd filters. https://github.com/PatWalters/rd\_filters, Accessed: Jan 14, 2019.

49. Bickerton, G. R.; Paolini, G. V.; Besnard, J.; Muresan, S.; Hopkins, A. L. Quantifying the chemical beauty of drugs. Nat. Chem. 2012, 4,90−98.

50. Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Adv. Drug Delivery Rev. 1997, 23,3−25.

51. EvoMol: a flexible and interpretable evolutionary algorithm – Jules Leguy

52. MolFinder: an evolutionary algorithm for the global optimization – Yongbeom Kwon

53. John J Irwin et al. “ZINC20—a free ultralarge-scale chemical database for ligand discovery”. In: Journal of chemical information and modeling 60.12 (2020), pp. 6065–6073.

54. DeLA-DrugSelf: Empowering multi-objective de novo design through SELFIES molecular representation - Domenico Alberga

55. Jan H Jensen. “A graph-based genetic algorithm and generative model Monte Carlo tree search for the exploration of chemical space”. In: Chemical science 10.12 (2019), pp. 3567–3572.

56. Emilie S Henault, Maria H Rasmussen, and Jan H Jensen. “Chemical space exploration: how genetic algorithms find the needle in the haystack”. In: PeerJ Physical Chemistry 2 (2020), e11.