**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 f ills 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. 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.

**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**

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 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. De novo design methods are often evaluated by their performance on standalone toy tasks, such as maximizing the quantitative estimate of drug-likeness(QED) [22]. 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 [23]. 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. 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 [24].

**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 [25]; 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) [26], 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 [27]. On the other hand, SELF-referencing Embedded Strings(SELFIES) [28], 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 opti mization (MOO) aims to coalesce signals from several weak scorers using data fusion concepts, such as Pareto optimality [29]. The design of effective MOO profiles is nontrivial and often makes use of normalization functions and scaling protocols when combining multiple objectives [30]. 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 [31,32]. 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].

**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. 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.

23. 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.

24. 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.

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

26. 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)

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

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

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

30. 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

31. 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

32. 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