Druglike: Applications of Proof-of-Optimization in Democratized Drug Discovery

Jason Sommer

jason@druglike.com

July 25, 2022

Abstract

In this paper, we introduce Druglike, a platform for democratizing the access, costs, and rewards of early-stage drug discovery. To offer a more competitive pricing schedule for cheminformatics compute, we propose a decentralized network of Task Providers, Solvers, and Validators tasked with executing and verifying virtual screening tasks. To maintain result integrity in this decentralized compute network while avoiding the inefficiencies of verification by replication, we introduce Proof-of-Optimization, a novel Proof-of-Work mechanism establishing consensus by validating and rewarding solutions to function optimization problems. We then show how this mechanism can be applied to the virtual screening problem, where Solvers can competitively search a chemical space for results demonstrating high binding affinity toward a given protein target.

Legal Disclaimer: Nothing in this White Paper is an offer to sell, or the solicitation of an offer to buy, any tokens. Druglike is publishing this White Paper solely to receive feedback and comments from the public. If and when Druglike offers for sale any tokens (or a Simple Agreement for Future Tokens), it will do so through definitive offering documents, including a disclosure document and risk factors. Those definitive documents also are expected to include an updated version of this White Paper, which may differ significantly from the current version. If and when Druglike makes such an offering in the United States, the offering likely will be available solely to accredited investors.

Nothing in this White Paper should be treated or read as a guarantee or promise of how Druglike’s business or the tokens will develop or of the utility or value of the tokens. This White Paper outlines current plans, which could change at its discretion, and the success of which will depend on many factors outside Druglike’s control, including market-based factors and factors within the data and cryptocurrency industries, among others. Any statements about future events are based solely on Druglike’s analysis of the issues described in this White Paper. That analysis may prove to be incorrect.
## Contents

1 Introduction 3  
   1.1 Virtual Screening  3  
   1.2 Platform Goals  3  

2 Virtual Screening Methods 4  
   2.1 Ligand-based  4  
   2.2 Structure-based  4  
   2.3 Molecular Docking  5  

3 Decentralized Computing 7  
   3.1 Costs  7  
   3.2 Verifiable Computing  7  
     3.2.1 Replication  8  
     3.2.2 Trusted Execution Environments  9  
     3.2.3 Zero-Knowledge Proofs  9  
   3.3 Limitations  10  

4 Optimization Solutions as Proof-of-Work 11  
   4.1 Discrete Functions  11  
   4.2 Continuous Functions  12  
   4.3 Challenges  12  

5 Proof-of-Optimization 15  
   5.1 Workflow  16  
   5.2 Domain Constraints  16  
   5.3 Reward Distribution Strategies  17  
     5.3.1 Highest-Scoring Solver  17  
     5.3.2 Multiple High-Scoring Solvers  17  
     5.3.3 Most Diverse Subset  17  
   5.4 Mining Pools  17  
   5.5 Edge Cases  18  
     5.5.1 Transaction Flooding  18  
     5.5.2 MEV Attacks  18  
   5.6 Cost Efficiency  18  
   5.7 Criteria Evaluation  19  

6 Democratizing Cost 20  
   6.1 Molecular Docking as Proof-of-Work  20  
     6.1.1 Domain Constraint  21  
     6.1.2 Reward Distribution Strategy  21  
   6.2 Virtual Screening as Proof-of-Work  22  
     6.2.1 QM/MM Approximations  22  
     6.2.2 Domain Constraint  23  
     6.2.3 Reward Distribution Strategy  23  
   6.3 Binding Affinity as a Partial Indicator  23  
   6.4 Other Applications  24
1 Introduction

The drug discovery and development process is a lengthy, complex, and costly interdisciplinary process that includes target identification, lead generation, lead optimization, assay development, clinical development, and eventual Food and Drug Administration (FDA) approval. An estimated average of $1.3 billion is spent on the successful discovery and development of a single FDA-approved drug \cite{WML20}. The average time span from the point of identifying a candidate to its FDA approval is around ten years \cite{TE09}.

Drug discovery, the first stage of this pipeline, first identifies a target protein, then searches chemical space to find a small group of candidate compounds (ligands) that bind to the target protein. This selection will then advance to in-vitro screening for further testing. Although search efforts are usually only concentrated on known chemical subspaces, these alone can become quite large. An estimated $10^{60}$ \cite{BMG96} compounds abide by Lipinski’s Rule of Five \cite{Lip+97}, a common heuristic for indicating oral bioavailability. The size of chemical space for all compounds up to 30 atoms is estimated to be $10^{20}–10^{24}$ compounds \cite{Ert03}.

1.1 Virtual Screening

Virtual screening is a widely used computational technique that leverages both machine learning and biomolecular simulation to quickly filter large sets of ligands to a smaller group of candidates that may demonstrate high binding affinity in subsequent in-vitro assays. Although high-throughput screening (HTS) robotics and other methods have been used primarily by the pharmaceutical industry for finding hits, virtual screening continues to gain adoption due to iteration capacity, scalability, speed, and ability to simulate compounds without having to synthesize them. Failing early, fast, and cheaply in the earlier steps offsets potentially wasted resources in later, more expensive and time-consuming steps such as pre-clinical development or clinical trials \cite{BC21}. Successful virtual screening campaigns have thorough documentation in the literature \cite{MS11}.

1.2 Platform Goals

Accurate, large-scale virtual screening capability is only readily available to larger pharmaceutical companies and specialized academic labs. Proprietary drug design software packages use costly product licensing models, including some that price by CPU core. Patent royalties are usually owned by larger pharmaceutical companies, and very little of the open-ended reward or equity flows back to innovators.

The Druglike platform is the culmination of many ideas on how to practically address these issues. The platform intends to democratize the access, costs, and rewards of early stage drug discovery. Platform goals include:

1. Democratizing cost by offering a more competitive pricing schedule for early stage cheminformatics compute that allows Task Providers and Solvers to negotiate between upfront price and royalty equity.

2. Democratizing access with a web-based tool suite for target identification, drug design, and constructing virtual screening workflows.

This paper mainly covers the aspects of democratizing compute cost. Future literature regarding other goals will soon follow.
2 Virtual Screening Methods

Existing compound collections only comprise a small region of chemical space. Unlike HTS, virtual screening (Figure 1) offers the ability to search outside the space of known drugs (A) in undiscovered yet potentially profitable regions of chemical space (B) [MS11]. Both ligand-based and structure-based virtual screening methods are used in this iterative process.

2.1 Ligand-based

Ligand-based virtual screening methods filter ligands strictly based on the characteristics of other ligands that bind to the target. Since no information about the target protein itself is used, ligand-based methods are most useful when no 3D structure of the target protein is available.

2.2 Structure-based

Structure-based virtual screening methods estimate the binding affinity of a candidate ligand from both the properties of the ligand and the geometry of the target protein. These target structures are usually procured through modifying existing structures from public databases such as RCSB PDB [Ber+00], through homology modeling.
2.3 Molecular Docking

Binding affinity is the quantitative strength of the binding interaction between a ligand and a target protein. This value is usually represented as the change in a system’s free energy associated with the binding process. Both binding affinity and ligand efficacy are required to predict the potency of a drug.

Molecular docking is a structure-based virtual screening method that simulates different positions and orientations (poses) of a ligand in free space, and predicts which poses will result in the highest binding affinity.

The historic analogy of the molecular docking problem is the "lock-and-key" challenge, where the algorithm attempts to find the correct orientation of the key that will correctly fit the lock [Fis94]. Since both the ligand and protein structures are flexible, a more appropriate analogy would be "hand-and-glove". Pose and affinity predictions are made for each ligand in a given chemical space, and the results are ranked by
binding affinity or other similar scores. The highest-scoring ligands then advance to the next stage of screening.

Most docking algorithms map pose and affinity prediction to a function optimization problem, where the function evaluates a pose and outputs a score or binding affinity estimate for the pose and target pair. As the function is optimized, the pose converges toward an energy minimized state. Mapping the docking problem to function optimization yields many solving approaches, including Tabu search [Bax+98], genetic algorithms [TC06], simulated annealing [GO90], and random forest classification [WZ17].
3 Decentralized Computing

The processing power required for molecular docking scales with the volume of the search space, the number of ligands screened, and the overall complexity of the ligands, normally correlated with the number of rotatable bonds. Recent academic campaigns have been able to virtually screen over one billion molecules [Ton+20]. Other screens of similar size run on cloud-provisioned hardware, often using tens of thousands of CPU cores for multiple days at a time [Gor+20]. On-premises supercomputers are also used for running large-scale virtual screening pipelines [Ach+20].

These large-scale screening projects require either special discounted deals with cloud compute providers, or prioritized academic access to high performance computing (HPC) resources. Smaller research groups may run screens on commodity cloud computing resources, but the profit margins on specialized, cloud-provisioned hardware usually constrain the size of the screen.

In the interest of democratizing the cost of early-stage drug discovery, we evaluate decentralized computing systems as a flexible yet economic competitor to both cloud and on-premises computing. Decentralized, distributed computing systems divide work over vast networks of individual processing nodes, where subsets of the network are capable of running independent of each other. This model also has the potential to both use and monetize the large supply of idle, internet-connected computing resources.

Major cryptocurrencies such as Bitcoin and Ethereum form some of the largest distributed computing systems in operation today. Even though their problem spaces are quite limited, at tens of millions of petaflops [BC22] they vastly exceed the computational power of even the largest supercomputers to date [TOP500].

Decentralized computing also has a proven scientific application in both biophysics and generic scientific compute. Folding@Home [SP00], a project aimed to help scientists develop new therapeutics for a variety of diseases by the means of simulating protein dynamics, has a combined computing power comparable to the world’s largest supercomputers [pub20].

3.1 Costs

Decentralized computing presents an attractive economic option for democratizing these large-scale tasks. If a decentralized compute network has a sizeable amount of worker nodes, they will form an economic market of compute supply, where nodes will effectively compete against each other to offer the lowest compute price that still justifies their participation in the network. This market price is often much less than the price point offered by cloud compute providers with much higher profit margins.

3.2 Verifiable Computing

Verifiable computing, the process by which a client distributes a unit of work to an untrusted worker while ensuring the full validity of the worker’s result [GGP10], is the primary challenge facing practical decentralized computing. Since the assumption must be made that bad actors will work against any protocol for profit, the robustness of the verifiable computing method will ultimately determine the eventual feasibility of the decentralized computing implementation.

Numerous solutions exist for verifiable computing, and they can be evaluated against a few independent criteria:
1. **Flexibility**: Are the supported types of distributed work inherently limited by the implementation? Flexible implementations of verifiable computing will process tasks regardless of whether the unit of work involves simulating a macromolecule, mining a Bitcoin block, predicting weather patterns, or planning a distribution route.

2. **Security**: How confident can the client be in the correctness of the results provided by the worker?

3. **Efficiency**: Can the client verify proof of work with significantly less effort than computing the work itself? How much effort is required by the network to compute the work compared to using a single trusted cloud compute instance with an adjustable profit margin?

4. **Portability**: Can the network’s nodes be run on standard commodity hardware?

5. **Decentralization**: How vulnerable is the implementation to a single point of failure?

With these criteria, we can examine the different available types of verifiable computing:

### 3.2.1 Replication

Verification by replication verifies remotely executed tasks by running the same task on \( N \) separate, uncoordinated workers and verifying that the results match. This approach is used by distributed task-agnostic compute providers (Golem Network \cite{G16} and iExec \cite{I18}) and task-specific compute providers (SETI@Home \cite{And+02}, Folding@Home \cite{SP00}, BOINC \cite{And19}, Gridcoin \cite{GRC}, and FoldingCoin \cite{FLDC}).

1. **Flexibility**: High: No limitations on the type of task or modifications to the task are required, since replication only requires a clone of the original computing environment.

2. **Security**: Variable: Redundancy can never guarantee a correct solution, only the probability of the solution being correct depending on \( N \) \cite{Sar01}. Increasing \( N \) increases the probability of the solution being valid, but decreases the efficiency of the task, which can exceed the cloud compute costs even with a modest amount of replication. Volunteer-compute applications such as Folding@Home and SETI@Home use a low redundancy (\( N \approx 2 \)) to counter fraud risk. Lower redundancy is acceptable since both these systems are volunteer-supported, and bad actors have less of a financial incentive to intervene, even though they occasionally attempt to do so \cite{SH, Beb+09}. Redundant verification also exposes a vulnerability for Sybil attacks, where the attacker gains the majority of influence in the network \cite{Dou02}.

3. **Efficiency**: Variable: Efficiency decreases as redundancy \( N \) increases. Since verification by replication is a trade-off between efficiency and security, the method will always compete against the potentially cheaper option of delegating the work once to a single trusted worker for a negotiable premium.

4. **Portability**: High: No extra hardware requirements are needed for running the worker node.

5. **Decentralization**: High: No limitations exist regarding who can participate in the redundant verification process.
3.2.2 Trusted Execution Environments

Trusted Execution Environments (TEEs) are isolated hardware environments that can run code protected from the host. TEEs preserve the integrity of the input and output data even if the enclave resides on an untrusted machine. Both Golem Network \cite{G16} and iExec \cite{I18} support TEEs as backends for CPU compute using the Intel SGX platform.

1. **Flexibility**: Medium: Even though some advances have been made in creating TEEs for GPU compute, the platforms are not as mature as those offered for secure CPU compute \cite{VVB18} \cite{Wan+20}. This excludes the ever-growing supply of GPU workload.

2. **Security**: High: TEEs can guarantee result integrity through remote attestation.

3. **Efficiency**: Medium: TEEs introduce a variable amount of performance overhead \cite{BBK18}.

4. **Portability**: Low: Workers must first acquire the TEE vendor’s hardware to host the enclave.

5. **Decentralization**: Low: Many TEE solutions rely on TEE hardware vendor services such as remote attestation \cite{Rab+21}.

3.2.3 Zero-Knowledge Proofs

A zero-knowledge proof (ZKP) is a mathematical method that can prove the truth of a statement without conveying any additional information about the statement to the verifier \cite{GMR85}. These have applications in verifiable computation by providing a way to prove the execution of the steps of a computation with the additional benefit of not having to reveal the final result.

ZKPs have already shown practical utility in cryptocurrencies by decongesting networks through transaction rollups \cite{ZKR22}. Recent developments in zk-STARKs \cite{Ben+18} have resulted in the development of zero-knowledge virtual machines (zkVMs) created to support verifying compute on instructions sets such as EVM and RISC-V \cite{R022}. A zkVM will generate a cryptographic proof/receipt of correct execution that can be verified, and could be used to run more generic types of compute.

1. **Flexibility**: Low: Software must either be cross-compiled with separate toolchains to add proof-generating capability \cite{Hea+22}, or be completely rewritten in a domain-specific language (DSL) that compiles to an arithmetic circuit \cite{CIR22}.

2. **Security**: High: ZKPs can guarantee result integrity \cite{GMR85}.

3. **Efficiency**: Low: ZKP efficiency can be measured in terms of the cost of both generating and verifying the proof. For more complex computational problems such as molecular docking or machine learning, generating the proof would still be much more costly than running the whole computation locally or on a high-margin cloud compute provider \cite{Ben+13} \cite{Hea+22}.

4. **Portability**: High: No extra hardware requirements are needed for running the prover or verifier. However, specialized hardware could accelerate the process of proof generation and verification in the future \cite{ZKH22}.

5. **Decentralization**: High: No limitations exist regarding who can participate in the proof-generation or verification processes.
3.3 Limitations

As of today, a practical, verifiable computing implementation that excels in all the listed criteria still remains out of reach. Since function optimization represents a significant portion of large-scale cheminformatics compute workload, we sacrifice the task flexibility requirement to satisfy the remaining criteria, and propose an efficient, secure, portable, and decentralized verifiable computing solution for function optimization problems based on existing Proof-of-Work mechanisms.
4 Optimization Solutions as Proof-of-Work

Proof-of-Work is a type of cryptographic proof by which a verifier can prove that a certain amount of computational effort has been expended by a prover. The primary usage of Proof-of-Work mechanisms in major cryptocurrencies is not necessarily to prove the expenditure of effort or even produce useful solutions from these large-scale computational efforts, but to establish a large cost barrier through high energy and hardware requirements to deter manipulation of data, specifically around token ownership.

Proofs of work are computationally hard to derive, but easy to verify by other parties. This asymmetry has solidified Proof-of-Work as a basis for consensus in major cryptocurrencies and other decentralized networks.

In this section, we’ll explore the changes required to direct the massive computational expenditures in a Proof-of-Work system toward solving function optimization problems.

4.1 Discrete Functions

To better understand the requirements for building a practical Proof-of-Work mechanism around function optimization, we will first describe Bitcoin’s Proof-of-Work method as the optimization of a discrete function [Nak09].

Let $S \in A$ be the set of all nonces in nonce space $A$ that, once appended to the hashed block $B$, creates a digest that hashes to a value not exceeding target $T$.

$$S = \{ n \in A \mid H(B + n) \leq T \}$$  \hspace{1cm} (1)

Given the indicator function $1_S(n)$ of the solution space $S$,

$$1_S(n) := \begin{cases} 1 & \text{if } n \in S, \\ 0 & \text{if } n \notin S. \end{cases}$$  \hspace{1cm} (2)

Let $F_b$ be the complement of the indicator function $1_S$.

$$F_b(n) = 1_{S^{\complement}}(n) = 1 - 1_S(n)$$  \hspace{1cm} (3)

With these definitions in place, Bitcoin mining can be represented as a coordinated, brute-force discrete minimization of $F_b$. See Figure 4.
Once represented this way, we notice that efficient minimization is virtually impossible for this discrete function. Since there is no useful gradient to evaluate, miners are reduced to using coordinated brute-force methods to solve, incidentally making this problem a robust method to establish predictable computational expenditure for Proof-of-Work. The first miner to find a nonce $n$ where $F_b(n) = 0$ is awarded the block. Because of the difficulty, miners can share both workload and reward by effectively coordinating a non-overlapping traversal of nonce space $A$, distributing rewards accordingly upon discovery of a minimum.

Note that function $F_b$ possesses multiple known global minima. It also has a discrete range $\{0, 1\}$ and domain. These properties will be helpful in contrasting this example with the minimization of a continuous function in the following subsection.

4.2 Continuous Functions

With the context of the previous example, consider a new continuous function $F_c(x)$ with a single unknown global minimum. See Figure 5.

Unlike $F_b$, $F_c$ has a non-discrete range. We can determine the minimum parameter for $F_b$ by comparing the result to 0. Unlike $F_b$, $F_c$ has no known global minimum to compare against, so we can only approximate the true global minimum through iteration.

Also note that $F_c$ is easier to optimize than $F_b$ since the gradient can be evaluated, and solvers can employ a larger range of optimization techniques to search for the global minimum.

4.3 Challenges

Optimizing a discrete function such as $F_b$ with a known, discrete range is a trivial task to incorporate within a Proof-of-Work mechanism. The implementation would be similar to Bitcoin, where the first solver to find the solution yielding the function’s global minimum would be proven to have expended work. The solver is also proven to
have solved the block by simply evaluating the function with the claimed solution, and verifying that the result matches the function’s global minimum. In the case of $F_b$, the correctness of the solution can be determined both exactly and immediately by comparing it to the function’s known global minimum, zero. Multiple solvers compete to provide the optimal solution and claim the reward, which would establish the network as a viable option for optimizing functions of this limited type.

However, most practical minimization problems involve continuous functions such as $F_c$, where the range, and consequently the global minimum, are completely unknown. It is impossible to determine the correctness of a solution exactly or immediately without knowledge of the global minimum. This also complicates who receives the reward.

Standard methods of minimization encounter the same problem, and choose instead to offer an approximation of the correct solution over a finite number of iterations that gradually advance the solution toward the global minimum. With this method, correctness of a solution can only be determined approximately since there is no absolute guarantee that the true global minimum was visited, and eventually due to the number of iterations required for successful convergence.

A similar strategy could be implemented with a Proof-of-Work mechanism, where solvers compete by publishing candidate solutions that minimize a function toward results approaching the global minimum. Rewards are distributed to the publisher of the minimal solution after a fixed time duration or a specific number of iterations. Given the economic assumption that solvers will be incentivized to publish increasingly minimized solutions for a chance to gain the reward, we can also assume that the eventual result will approach the global minimum. This approach, along with other factors and edge cases, is formalized as Proof-of-Optimization in Section 5.

It is also possible to transform a continuous function minimization problem to one that finds a solution that only achieves a predefined threshold rather than the global minimum. Although this threshold condition would indicate both the exact and immediate correctness of the solution, this would complicate the process of choosing an appropriate threshold value. Setting the threshold too low would not reveal anything
about the global minimum. Setting the threshold too high would exceed the global minimum, preventing any reward from being distributed at all.
5 Proof-of-Optimization

Proof-of-Optimization is a system where independent Solvers are rewarded for providing the most optimized input $x$ for an arbitrary function $F(x)$.

$$y = \min_{x \in A} F(x)$$

(4)

More formally, Proof-of-Optimization is a novel Proof-of-Work method where the work is the minimization of a continuous or discrete function $F(x)$ over solution space $A$, the proof is the solution $x \in A$ that minimizes $F(x)$, and the verification of proof $x_n$ is determined by whether $\forall x \in A_c, F(x_n) \leq F(x)$, where $A_c \in A$ is the set of all candidate solutions provided by solvers competing for the verification reward $r$ after a certain duration $t$. Since $\max(F) = -\min(-F)$, Proof-of-Optimization can also maximize $F$ by minimizing $-F$ and negating the result.

This retains the characteristic Proof-of-Work asymmetry, since deriving the proof (optimizing $F$) is much more computationally difficult than verifying the proof (comparing candidate solutions in $A_c$).

The practicality of Proof-of-Optimization rests on the ability to incentivize independent Solvers to populate the set of candidate solutions $A_c$. We propose a blockchain-based implementation of Proof-of-Optimization, where a distributed ledger stores records of which proof solutions belong to which Solvers. Smart contracts allow secure distribution of rewards to the Solver who owns the verified proof.
5.1 Workflow

1. Task Provider publishes an Optimization Task. The Optimization Task describes function $F$, constrained domain $C \in A$ (Section 5.2), optimization type $t$ (minimization, maximization), reward $r$, reward distribution strategy $R$ (Section 5.3), and duration $d$.

2. Smart contract transfers reward $r$ from Task Provider to escrow account.

3. Solver optimizes $F(x)$ with additional context from the constrained domain $C$.

4. Solver publishes a candidate solution $x$ to the Optimization Task’s candidate solution set $A_c$.

5. For each candidate solution $x$, a pool of validators checks that the solution first belongs to the constrained domain $x \in C$, then calculates the result value $y = F(x)$. Once the validators individually calculate the result and establish consensus, the solution and result pair $(x, y)$ is published to the Optimization Task’s validated solution set $A_v$.

6. After duration $d$, the smart contract allows any Solver or Task Provider to invoke a method that distributes reward $r$ to winning Solvers according to the validated solution set $A_v$, optimization type $t$, and reward distribution strategy $R$.

5.2 Domain Constraints

The Task Provider may choose to optimize a function $F$ within a constrained subset of its domain $A$. The Task Provider always has the option to modify function $F$ to penalize solutions outside the desired domain constraint, but this gives the Solver no information about the constraint and leads to inefficiency during optimization. A more efficient method is to define a constrained domain $C \in A$ within the Optimization Task. If no constraint is desired, the Task Provider can set $C = A$. 
5.3 Reward Distribution Strategies

The Task Provider may want to choose a different reward distribution strategy based on the objective of the Task. For example, Task Providers may choose to reward a broader survey of the optimization landscape versus rewarding a strict search for the global minimum. We can express this variation with a reward distribution strategy $R$, a function that accepts a solution $x_n$ within the set of verified solutions $A_v$ and outputs the Solver’s share of reward $r$ as a fraction of one:

$$\sum_{x \in A_v} R(x) = 1$$  \hspace{1cm} (5)

Let $X$ be the set of verified solutions $A_v$ sorted in direction $t$ by value $y$ with a preference to the earlier provided solution in case of a duplicate. We can define the following simple reward distribution strategies:

5.3.1 Highest-Scoring Solver

Reward the single Solver of the optimal solution, preferring the first submitted solution in the event of a tie.

$$R_t(X_i) = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}$$ \hspace{1cm} (6)

5.3.2 Multiple High-Scoring Solvers

Reward the top $n$ Solvers with a share proportional to their relative ranking.

$$R_{mt}(X_i) = \frac{1}{i \sum_{j=1}^{n} 1/j}$$ \hspace{1cm} (7)

5.3.3 Most Diverse Subset

Reward the most optimized solutions along with an extra bias toward diverse mid-range solutions according to a function $D(x_i, x_j) \rightarrow \mathbb{R}^+$ that outputs the distance between solutions in an arbitrary space. We can define a most-diverse-subset (MDS) helper function that outputs a reward distribution strategy which rewards $n$ solutions, with a controllable bias factor $b$ that determines the proportion of the reward granted to globally optimal solutions versus diverse, locally optimal solutions according to the distance function $D$.

$$R_d = \text{MDS}(D, n, b)$$ \hspace{1cm} (8)

This strategy would reward Solvers who discover significant local minima, giving the Task Provider more insight into the optimization landscape of $F$. The implementation of this reward distribution strategy will be discussed in future documentation.

5.4 Mining Pools

Unless they are centrally coordinated, Solvers will inevitably traverse and evaluate overlapping regions of constrained domain $C$. Mining pools are organizations of Solvers that coordinate to evaluate non-overlapping partitions of the constrained domain $C$ for a shared reward.
5.5 Edge Cases

5.5.1 Transaction Flooding

Transaction flooding occurs when an attacker spams a blockchain network with a large volume of transactions. This method is usually used to attempt a distributed denial-of-service (DDoS) attack against the nodes in a blockchain network. The attack may also be lucrative in Proof-of-Optimization, where a Solver may first publish a highly-optimized solution, then subsequently flood the validators with low quality solutions to prevent them from potentially approving any competing solutions before reward distribution.

To combat this risk, each candidate solution $x$ published by a Solver could also include a result upper bound claim $y_b$ along with a collateral token amount. If the validator determines that the actual solution $y = F(x)$ exceeds the bound ($y > y_b$), the candidate solution is deemed invalid, and the collateral is slashed. If the solution remains within the bound ($y \leq y_b$), the solution is considered valid and the collateral is returned to the Solver. Validators would require a minimum collateral to process a candidate solution, and they would prioritize validation of optimal solutions by first validating candidates with lower values of the claimed upper bounds $y_b$.

5.5.2 MEV Attacks

Maximal/miner extractable value (MEV) refers to the maximum value that can be extracted from block production in excess of the standard block reward and gas fees by including, excluding, and changing the order of transactions in a block [MEV22]. In a Proof-of-Optimization system, validators and other Solvers may use MEV extraction techniques to observe candidate solutions, and resubmit copies of them with higher gas prices to front-run the original Solver, effectively capturing the reward.

This vulnerability can be mitigated using commitment schemes, where the Solver first publishes (commits) the encrypted candidate solution. Validators will be unable to evaluate the encrypted candidate solutions, but will be able to achieve consensus on the order of the encrypted candidate solutions within the block. Upon the next block, the Solver would then freely publish (reveal) the unencrypted candidate solution, and the validators would use encryption to locate and verify the solutions based on their order within the last block. This strategy is available for any chains having a deterministic leader schedule.

5.6 Cost Efficiency

The potential cost savings of Proof-of-Optimization ($c_p$) can be demonstrated by showing $c_h < c_p \ll c_c$ where $c_h$ is cost of local hardware, and $c_c$ is cost of cloud compute.

Let optimization and validation work be $O$ and $V$ respectively. Let $k$ be the overhead cost factor, which we will assume as constant among the different types of compute.

Given $n$ validators, we can represent the cost of Proof-of-Optimization:

$$c_p = k(O + nV) \quad (9)$$

For executing on local hardware:

$$c_h = kO \quad (10)$$
For executing on cloud compute providers with net profit margin $m$:

$$c_c = \frac{kO}{1 - m} \quad (11)$$

1. Since $c_p > c_h$ if $nV > 0$, the cost of Proof-of-Optimization will surpass the cost of local hardware ($c_p > c_h$) as either the number of validators $n$ increases or the work required for validation $V$ increases.

2. Since $c_c > c_p$ if $\frac{O}{V} > \frac{n}{m} - n$, we see that cost of Proof-of-Optimization will decrease compared to cost of cloud compute ($c_p \ll c_c$) as the compute complexity required for optimization becomes far greater than that required for validation ($O \gg V$), as net profit margin $m$ increases, or as the number of validator nodes $n$ decreases.

3. Assuming a cloud compute profit margin of 50% ($m = 0.5$), and an optimization process that is 10 times more difficult than validation ($\frac{O}{V} = 10$), we will have cost savings using Proof-of-Optimization ($c_p < c_c$) as long as the number of validator nodes remains under 10 ($n < 10$).

Note that Proof-of-Optimization does not eliminate all redundancy ($nV$), but only uses redundancy on the validation step to establish consensus.

### 5.7 Criteria Evaluation

Proof-of-Optimization can be evaluated against the previously defined criteria:

1. **Flexibility**: Low: Only function optimization problems can be solved.

2. **Security**: High: Proof-of-Stake methods can ensure the slashing of validator collateral if the validator behaves dishonestly.

3. **Efficiency**: Medium-High: Not high since redundant verification requires more total compute versus unverified local compute, but not medium since there’s thorough opportunity for competing against the cloud compute price point.

4. **Portability**: High: No extra hardware requirements are needed for proving or verifying.

5. **Decentralization**: High: No limitations exist regarding who can prove or verify.
6 Democratizing Cost

To democratize the cost of select types of early-stage cheminformatics compute, we can map both the molecular docking process and even the entire virtual screening process to optimization problems solvable by Proof-of-Optimization.

6.1 Molecular Docking as Proof-of-Work

The general molecular docking process can be expressed as a function minimization problem. Let ligand pose \( p \) represent a vector of parameters including position of the root atom \((x, y, z)\), the quaternion rotation of the root atom \( q \), and torsional components \( t_n \) for each rotatable bond \( n \) in the ligand. See Figure 7.

\[
p = (x, y, z, q_1, q_2, q_3, q_4, t_1, t_2, ..., t_n)
\] (12)

Figure 7: Components of ligand pose \( p \)

Molecular docking uses a scoring function \( S(p, t) \) that estimates the binding affinity between a single ligand pose \( p \) and target protein conformation \( t \). The docking process is the minimization of the scoring function \( S \) over the ligand pose vector space \( P_l \) and the flexible target conformation space \( T \) to find minimum free energy \( g \).

\[
g = \min_{p \in P_l, c \in T} S(p, c)
\] (13)

In terms of Proof-of-Optimization, solution space \( A \) represents all possible ligand pose and target conformation combinations:

\[
A = P_l \times T = \{(p, c) \mid p \in P_l \land c \in T\}
\] (14)

The solution \( x \in A \) will contain both the ligand pose and target conformer:

\[
x = (p, c)
\] (15)

The scoring function \( S \) is minimized function \( F \):

\[
F(x) = S(x_p, x_c)
\] (16)

The result of the minimization is the lowest observed free energy, and the optimal input arguments are the docked ligand pose and target protein conformation.

Scoring functions such as \( S \) usually depict a complex energy landscape, which makes molecular docking an ideal candidate for an advanced minimization problem. An example scoring function from AutoDock 4 can be found in Equation 17.
\[
\Delta G = \Delta G_{vdW} \sum_{i,j} \frac{A_{ij}}{r_{ij}^n} - \frac{B_{ij}}{r_{ij}^6} + \Delta G_{\text{hbond}} \sum_{i,j} E(t) \left( \frac{C_{ij}}{r_{ij}^n} - \frac{D_{ij}}{r_{ij}^6} + E_{\text{hbond}} \right) + \Delta G_{\text{elec}} \sum_{i,j} \frac{q_i q_j}{r_{ij}} + \Delta G_{\text{tor}} + \Delta G_{\text{sol}}
\]

Because of the complexity of scoring functions, molecular docking is an ideal application of Proof-of-Optimization, since it remains easier to verify a pose’s binding affinity compared to finding the optimal pose.

6.1.1 Domain Constraint

Molecular docking search boundaries can be implemented with domain constraints \( C \in A \) on the root atom position \((x, y, z)\).

Simple applications include restricting the search space to a box or sphere, but more complex methods include isolating the search space to multiple binding sites, or excluding certain known binding sites altogether for further study.

6.1.2 Reward Distribution Strategy

Using the Highest-Scoring Solver strategy (Section 5.3.1) will optimize for the ligand pose with the highest binding affinity.

To reward Solvers that provide more diverse candidate poses located at both global and local minima in the energy landscape, we can use the Most Diverse Subset strategy (Section 5.3.3) with root-mean-square deviation (RMSD) as the solution distance function \( D \). RMSD is used to measure the difference between two poses of the same ligand molecule docked on the same target protein:

\[
\text{RMSD} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \delta_i^2}
\]

\( \delta_i \) is the distance between an atom \( i \) and either a reference structure or the mean position of the \( N \) equivalent atoms.

Using RMSD as the distance function \( D \),

\[
D(x_1, x_2) = \text{RMSD}(x_{1p}, x_{2p})
\]

The Most Diverse Subset strategy can be used for reward distribution:

\[
R = \text{MDS}(\text{RMSD}, n, b)
\]
6.2 Virtual Screening as Proof-of-Work

Proof-of-Optimization can also be mapped to problems of a much larger scope. Consider the entire virtual screening process itself as a free energy optimization over the combined space of all possible ligands and their poses, rather than just all poses for a single ligand. From a large group of ligands, the optimized solution will select the ligand and pose with the highest demonstrated binding affinity.

Standard docking methods use force field (FF) approximations to simulate interaction between a ligand and protein target during the iteration of poses for a single ligand. These FF approximations are ideal for the docking process due to an acceptable tradeoff between speed and accuracy, but successful hits normally enter a next round of slower, much more accurate free energy prediction using molecular dynamics (MD) simulations at the combined quantum-mechanical/molecular mechanics (QM/MM) level. QM/MM MD simulations usually involve simulating a ligand and protein complex for several nanoseconds at a resolution of a few femtoseconds to achieve an accurate affinity estimate that accounts for the flexibility of both the ligand and the target protein along with the quantum mechanical effects present in the interaction [OR17].

General virtual screening workflows can be mapped to a scoring function optimization problem over a group of ligands.

Let $P$ be the set of all poses for all ligands in the set $L$:

$$ P = \{ p \in P_l \mid l \in L \} $$ (21)

In terms of Proof-of-Optimization, solution space $A$ represents all combinations of ligands, their poses, and target conformation combinations:

$$ A = P \times T $$ (22)

The solution $x \in A$ will contain both the ligand pose and target conformer:

$$ x = (p, c) $$ (23)

The minimized function $F(x)$ is a QM/MM free energy estimation function $S_{qm}$ which outputs the binding affinity between the ligand and protein target after a QM/MM simulation with duration $d$ and resolution $r$.

$$ F(x) = S_{qm}(x_p, x_c, d, r) $$ (24)

Note that the high-accuracy scoring function is not limited to QM/MM, and is assignable based on the needs of the Task Provider. Solvers will optimize over the combined space of all possible ligands and their poses $A$ for the highest binding affinity predicted by the high-accuracy scoring function $F(x)$.

6.2.1 QM/MM Approximations

QM/MM methods apply an appropriate level of quantum chemistry theory to the binding region of the simulation, while applying a cheaper molecular mechanics (MM) force field method to the remainder. GPU-accelerated software such as OpenMM [Eas+17], AMBER [SCW13], and Gromacs [Abr+15] is used to perform these simulations.

Since running a QM/MM simulation $S_{qm}$ for each iteration of the optimization process would be prohibitively slow, Solvers may choose to optimize an approximate
scoring function $S_a$ that uses standard MM force fields, empirical methods, or machine learning [Jon+21][Gen+20].

$$S_a \approx S_{qm}$$  \hspace{1cm} (25)

Reward opportunities would encourage Solvers to build and use faster, more efficient versions of $S_a$.

### 6.2.2 Domain Constraint

High binding affinity alone is insufficient for indicating the true druggability of a compound. Other ligand-based heuristics such as ADMET profiling, excluded SMARTS matches, and PAINS-removal are important for filtering larger chemical spaces into a smaller group of advanceable compounds. To prevent solvers from minimizing unwanted compounds that fail these pharmacological heuristics, they can be encoded as a part of the domain constraint $C$ within the Optimization Task.

Most virtual screening tasks will default to using ligand-based heuristics as the domain constraint. These include druggability indicators such as Lipinski’s Rule of Five, ADMET profiling, as well as more complex heuristics that prevent Solvers from exploiting the system by providing highly optimized yet useless compounds.

Using ligand-based heuristics alone will allow Solvers to provide any arbitrary compound as a solution, whether it was discovered in a large ligand library, or even if it was manually designed by the Solver. Task Providers will have the option to further constrain the domain to a fixed set of compounds from their own libraries.

When the domain is constrained to a fixed set of compounds, Solvers will refrain from published candidate solutions for any of the compounds in the set classified as low-affinity, since they do nothing to help the Solver achieve the reward. The Task Provider will be unable to get an accurate score for every one of the compounds specified. In this case, the Task Provider may choose to split the virtual screening task into $N$ Molecular Docking problems (Section 6.1) for each of the $N$ compounds they want to screen. This way, Solvers will have an incentive to screen each compound for a potential reward. This is important for providing the Task Provider an exhaustive evaluation of the entire library, especially in cases such as toxicity studies where the aim is to check for the absence of binding affinity for a particular ligand.

### 6.2.3 Reward Distribution Strategy

To prevent multiple Solvers from immediately converging on the first optimal chemotype found, some incentive must be given for providing a more diverse yet optimal cohort of ligands. The Most Diverse Subset strategy (Section 5.3.3) can be used with Tanimoto distance $J$ as the solution distance function $D$ to reward dissimilar compound solutions.

$$D(x_1, x_2) = J(x_{1l}, x_{2l}) = J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$  \hspace{1cm} (26)

$$R = \text{MDS}(J, n, b)$$  \hspace{1cm} (27)

### 6.3 Binding Affinity as a Partial Indicator

The virtual screening process is designed to predict good ligands, but not necessarily good drugs [BC21]. Strong evidence of binding affinity from QM/MM simulation is a
helpful indicator that can eliminate leads that are false positives, but it still gives little information on how the hit will perform in the later stages of the drug development process.

The same market mechanism that rewards accurate predictions of binding affinity through a market of competitive Solvers could operate at a much larger scale within the drug discovery and development process to reward all participants in the chain of discovery for a profitable drug candidate. This idea will be discussed further in future documentation.

6.4 Other Applications

Proof-of-Optimization is not only limited to solving biopharmaceutical problems, but can also find applications in other fields that rely on function optimization, including physics, engineering, and logistics. The Druglike platform intends to host a distributed computing network with the first practical implementation of Proof-of-Optimization, focusing first on cheminformatics workloads, and later focusing on eventual support for other types of scientific compute.
References


[Acharya+20] A. Acharya et al. “Supercomputer-Based Ensemble Docking Drug Discovery Pipeline with Application to Covid-19”. In: *Journal of Chemical Information and Modeling* 60.12 (Dec. 2020), pp. 5832–5852. ISSN: 1549-9596. DOI: [10.1021/acs.jcim.0c01010](https://doi.org/10.1021/acs.jcim.0c01010). URL: [https://doi.org/10.1021/acs.jcim.0c01010](https://doi.org/10.1021/acs.jcim.0c01010) (visited on 07/02/2022).


