Innovations in Computer Science 2010

# Computational Complexity and Information Asymmetry in Financial Products*

**(Extended Abstract)**

Sanjeev Arora[1] Boaz Barak[1] Markus Brunnermeier [2] Rong Ge [1]
[1]Department of Computer Science and Center for Computational Intractability, Princeton University
[2]Department of Economics and Bendheim Center for Finance, Princeton University
arora@cs.princeton.edu barak@cs.princeton.edu markus@princeton.edu rongge@cs.princeton.edu

**Abstract:**
This paper introduces notions from computational complexity into the study of financial derivatives. Traditional economics argues that derivatives, like CDOs and CDSs, ameliorate the negative costs imposed due to *asymmetric information* between buyers and sellers. This is because securitization via these derivatives allows the informed party to find buyers for the information-insensitive part of the cash flow stream of an asset (e.g., a mortgage) and retain the remainder. In this paper we show that this viewpoint may need to be revised once computational complexity is brought into the picture. Assuming reasonable complexity-theoretic conjectures, we show that derivatives can actually amplify the costs of asymmetric information instead of reducing them. We prove our results both in the worst-case setting, as well as the more realistic average case setting. In the latter case, to argue that our constructions result in derivatives that "look like" real-life derivatives, we use the notion of *computational indistinguishability* a la cryptography.

**Keywords:** intractability; economics; financial derivatives; asymmetric information; lemon cost; dense subgraph

## 1 Introduction

A *financial derivative* is a contract entered between two parties, in which they agree to exchange payments based on the performance or events relating to one or more underlying assets. For instance, the contract may specify that the buyer pays the seller \$1M if the DOW is above $11,000$ exactly 1 year after the contract date. In recent years the market for derivatives has grown tremendously, both in volume and sophistication. The total volume of trades dwarfs the world's GDP. Of special interest in this paper are derivatives such as *collateralized debt obligations* (CDOs) used in *securitization* of debt, which transformed the financial industry over the last three decades. By bundling together risky assets like consumer mortgages or credit card debt, they create new classes of assets that are — assuming a "law of large numbers" applies—much less risky.

The explosive growth of derivatives has attracted criticism. Warren Buffet famously called derivatives "financial weapons of mass destruction," and many believe derivatives played a role in enabling problems in a relatively small market, U.S. subprime lending, to cause a global recession. (See Section 2 for more background on financial derivatives and [7, 8] for a survey of the role played by derivatives in the recent recession.) Critics suggest that derivatives should be regulated by a federal agency similar to FDA or USDA. Opponents of regulation counter that derivatives are contracts entered into by sophisticated investors in a free market, and play an important beneficial role that would be greatly harmed by a slow-moving regulatory regime akin to that for medicinal drugs and food products.

From the viewpoint of economic theory, derivatives can be beneficial by "completing the market" and by helping ameliorate the effect of *asymmetric information*. The latter refers to the fact that securitization via derivatives allows the informed party to find buyers for the information-insensitive part of the cash flow stream of an asset (e.g., a mortgage) and retain the remainder. DeMarzo [9] suggests this beneficial effect is quite large. He shows that even though one would expect that informed sellers would be able to cheat their buyers by suitable "cherry picking" based

*Full version of this extended abstract available from `http://www.cs.princeton.edu/ rongge/`. Research supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797, 0830673, 528414, US-Israel BSF grant 2004288, Sloan Foundation, and Packard Foundation.

upon their hidden information, in fact derivatives like CDOs protect the buyers.

The practical downside of using derivatives is that they are *complex* assets that are difficult to price. Studies suggest that valuations for a given product by different sophisticated investment banks can be easily 17% apart [5] and that even a single bank's evaluations of different tranches of the same derivative may be mutually inconsistent [11]. Many sources for this complexity have been identified, including the complicated structure of many derivatives, the sheer volume of financial transactions, the need for highly precise economic modeling, the lack of transparency in the markets, and more. Recent regulatory proposals focus on improving transparency as a partial solution to the complexity.

This paper shows that the seeming difficulty of pricing is quantifiable using computational complexity, and has a strong bearing on the issue of asymmetric information. From a distance, such results should not look surprising to computer scientists. Consider for example a derivative whose contract contains a 1000 digit integer $n$ and has a nonzero payoff iff the unemployment rate next January, when rounded to the nearest integer, is the last digit of a factor of $n$. A relatively unsophisticated seller can generate such a derivative together with a fairly accurate estimate of its yield (to the extent that unemployment rate is predictable), yet even sophisticated buyers like Goldman Sachs would have no idea what to pay for it since they do not know how to factor $n$. This example shows both the difficulty of pricing arbitrary derivatives and the possible increase in asymmetry of information via derivatives. It suggests that derivative contracts could contain information that is in plain view yet cannot be understood with any foreseeable amount of computational effort. This can be viewed as an extreme case of bounded rationality [14] whereby even the most sophisticated investment banks cannot be fully rational since they do not have unbounded computational power.

Of course, the above "factoring derivative" is far from any contract that is currently traded on derivatives market, so such a hardness result is not useful. Popular derivatives in securitization such as CDOs involve *threshold functions* (see Section 2). We show that interesting problems involving derivative pricing are NP-hard for even these classes of derivatives, and in fact are NP-hard to approximate (Section 3). The intractability of approximation arises in cases where there is a small amount of asymmetric information — the seller knows a little bit more than the buyer about the underlying assets.

While the above hardness result hints at the difficulty of designing accurate pricing algorithms, it does not —like all NP-hardness results— rule out this endeavor on "real-life" instances.

In the finance industry as well as finance theory it is customary to consider assets whose yields are distributed according to standard models like *Gaussian copula*. From a computer science viewpoint this implies moving to the setting of average-case complexity. The model used in this extended abstract is a very simple subcase of the Gaussian copula, where yields are either 0 or 1. Our most important result shows that the pricing problem in our simple model are at least as hard as the *planted dense subgraph* problem, even in these settings. (Note that since this is a hardness result, if it holds in simpler models it also extends for any more complicated model that contains the simpler model as a subcase.) Furthermore, we argue that the hard instances "look indistinguishable" from real-life instances that are randomly generated.

**High level idea:** The high level idea in our main result is that everyday derivatives like CDOs are defined using threshold functions on various subsets of the asset pools. Our result is related to the well known fact that random election involving $n$ voters can be swung with significant probability by making $\sqrt{n}$ voters vote the same way. *Private information* for the seller can be viewed as a restriction of the input distribution known only to the seller. The seller can structure the derivative so that this private information corresponds to "swinging the election." What is surprising is that a computationally limited buyer may not have any way to distinguish such a *tampered* derivative from untampered derivatives. Formally, the indistinguishability relies upon the conjectured intractability of the *planted dense subgraph* problem.[1] This is a well studied problem in combinatorial optimization (e.g., see [6, 12, 15]), and the planted variant of it has also been recently proposed by Applebaum et al. [3] as a basis for a public-key cryptosystem.

**Akerloff's notion of lemon costs.** Akerloff's classic 1970 paper [2] gives a simple framework for quantifying asymmetric information. The simplest setting is as follows. You are in the market for a used car. A used car in working condition is worth \$1000. However, 20% of the used cars are *lemons* (i.e., are useless, even though they look fine on the outside) and their true worth is \$0. Thus if you could pick a used car at random then its expected worth would be only

[1] Note that debt-rating agencies such as Moody's or S&P currently use simple simulation-based approaches [20], and hence may fail to detect tampering even in the parameter regime where the densest subgraph can be solved in polynomial-time using, say, semidefinite programming.

$800 and not $1000. Now consider the seller's perspective. Suppose sellers know whether or not they have a lemon or not. Then a seller who knows that his car is not a lemon would be unwilling to sell for $800, and would exit the market. Thus the market would feature only lemons, and nobody would buy or sell. Akerloff's paper goes on to analyze reasons why used cars do sell in real life. We will be interested in one of the reasons, namely, that there could be a *difference* between what a car is worth to a buyer versus a seller. In the above example, the seller's value for a working car will have to be $200 less than the buyer's in order for trade to occur. In this case we say that the "lemon cost" of this market is $200. Generally, the higher this cost, the less efficient is the market.

**Cost of complexity.** We will define a notion of "lemon cost" for derivatives. When sellers have more information than buyers, they have an incentive to structure the derivatives to their advantage. The lemon cost is the maximum advantage they can gain (see Section 2.1) on their entire bundle of assets. We quantify the *cost of complexity* by comparing the lemon cost in the case that the buyers are computationally unbounded, and the case that they can only do polynomial-time computation. We will show that there is a significant difference between the two scenarios.

Note that the lemon issue for derivatives has been examined before. It is well-recognized that since a seller is more knowledgeable about the assets he is selling, he may design the derivative advantageously for himself by suitable cherry-picking. However it was believed (and shown in the above-mentioned work of DeMarzo) that the effects of such cherry-picking can be mitigated. This belief is questioned by our paper.

Could one tackle the tempering issues raised here by instituting a *Lemon Law* for derivatives, guaranteeing a way to "roll-back" an agreement if an irregularity is discovered? We suggest (see Section 5.1) a surprising answer: in many models, even the problem of detecting the tampering *ex post* may be intractable. In contrast, we also mention some preliminary results (see Section 6) suggesting that one could mitigate the problems we identify by using certain exotic derivatives whose design (and pricing) is influenced by computer science ideas. Though these are provably tamper proof in our simpler model, it remains to be seen if they can find economic utility in more realistic settings.

**Companion paper:** This paper is geared to a CS audience and proves simple cases of the result — for binary CDOs only—that showcases the essential computer science ideas. There is a companion paper geared to economics audiences that quantifies the economic effects of complexity more carefully. The hardness results are proved in more realistic models of derivatives that involve *tranching*, and they lead to classification of different derivative families according to the "cost of complexity." The asset distributions involve industry-standard models, and the lemon issue is studied in a DeMarzo-like model of asymmetric information, which contrasts the results with DeMarzo's and thus may be interesting to economists.

## 2 Definitions and Finance Background

A derivative is a contract entered between two parties to exchange payments based on some events to underlying assets. Derivatives are extremely general: the events can be discrete events such as default or change in credit rating, as well as just market performance. The assets themselves can be a company, a loan, or even another derivative. Neither party has to actually own the assets in question. The payment structure can also be fairly complex, and can also depend on the timing of the event. See the book [16] for a general treatment of derivatives and [17] for a practical aspects of evaluating the kind of derivatives discussed below. The papers [7, 8] discuss the role derivatives played in bringing about the recent economic crisis. In Section 4.4 we compare the features of our model with the kind of derivatives used in practice.

Formally, a derivative may be viewed as a function $f(X_1, X_2, \ldots, X_s)$ that maps a vector of economic variables $X_1, X_2, \ldots, X_s$ to a real number, which represents the *payment* (possibly negative) from the first party to the second. This function is specified in the derivative's contract, which can run into hundreds of pages. Economists typically think of the $X_i$'s as stochastic variables (possibly dependent). Thus the fair "price" of this derivative for a risk averse buyer is $\mathrm{E}[f(X_1, X_2, \ldots, X_s)]$. For example, in a *credit default swap (CDS)*, Party A "insures" Party B against default of a third Party C, by promising to pay amount $M$ if $C$ defaults in one year. This contract may be viewed as specifying a function $f(X_1)$ where $X_1$ is an indicator random variable for the event "C defaults in one year" and $f : \{0, 1\} \to \Re$ where $f(0) = 0, f(1) = M$.

In the *derivative pricing* problem, we are given a description of $f$, and joint distribution of the underlying variables $X_1, X_2, \ldots, X_s$, and need to compute $\mathrm{E}[f(X_1, X_2, \ldots, X_s)]$. While computing the expectation exactly is $\sharp\mathbf{P}$, if $f$ is efficiently computable in a bounded domain, and the joint distribution of

$X_1, X_2, \ldots, X_s$ is sampleable, then one can approximate this expectation up to arbitrary inverse polynomial accuracy using the *Monte Carlo* method (i.e., repeatedly sample values for $X_1, X_2, \ldots, X_s$ and empirically estimate $E[f(X_1, X_2, \ldots, X_s)]$), and this is a popular approach in practice.

We will be interested in derivatives that arise in debt *securitization*, where consumer debt such as mortgages are packaged into new products with lower risk. A simple example is a *collateralized debt obligation (CDO)*. For simplicity suppose the seller owns $D$ mortgages whose returns are iid Bernoulli variables that are equally likely to be 1 or 0. Individually, these may be too risky for risk-averse investors such as pension funds. Viewed as a bundle, these can be viewed as a single asset whose distribution is $B(D, 1/2)$ and thus approximately gaussian when $D$ is moderately large. This entire bundle is split into two parts according to a threshold $\alpha \in [0, 1]$. The first part, called the *senior tranche* is a claim to the first $\alpha D$ units of income from the assets. Typically $\alpha$ is less than the expected fraction of assets with a return (i.e. $1/2$ in this case), and so with very high probability the senior tranche's return will be exactly $\alpha D$. For this reason senior tranches were often considered investment grade assets and given very high ratings such as AAA (the highest possible rating reserved only for the safest of investments). The second part, called the *junior tranche*, is a claim to the rest of the income. Obviously, the junior tranche has much higher variance than the senior tranche. The sellers often retain this tranche as a sign of confidence in the underlying assets. (We note that the above is a highly simplified description. Often the pool is split into more than two tranches.)

From a computer science viewpoint, a CDO is a *threshold function* (aka perceptron). It is not uncommon to also find $CDO^2$, which are CDOs of CDOs. These can be viewed as depth 2 threshold circuits. Higher depth circuits such as $CDO^3$ also exist but are less common. For all of those approximate pricing can be done via Monte Carlo simulation, while it's $\sharp\mathbf{P}$-complete to compute the exact expectation for higher depth circuits.

In this paper we are concerned with situations involving *asymmetric information*, where the seller may have more accurate information about the distribution of $X_i$'s than the buyer. For example, a bank that creates the above CDO could know that certain mortgages will default with probability higher than $1/2$. This problem of asymmetric information has been recognized before, and the common wisdom is that the senior tranche isolates the buyer from its effect, and hence allows them to invest in the *information insensitive* part of the pool, where they are not at a disadvantage compared to the seller. This paper seeks to question this received wisdom.

In this version geared to a CS audience, we study these issues with respect to *Binary CDOs*, which is a discrete version of the above, where the senior tranche gets $\alpha D$ as long as the total yield of $D$ assets is above $\alpha D$, and gets nothing otherwise. The junior tranche gets the rest of the income. While binary CDOs are not as common in practice, they capture all the essential elements of the more general case, which is presented in the companion paper to this paper.

### 2.1 Quantifying Asymmetric Information, Lemon Costs and Cost of Complexity

We borrow notions from Akerloff's work on lemons to quantify the cost of asymmetric information between seller and buyer. For simplicity this definition assumes that assets take values in $\{0, 1\}$. A *lemon* is an asset whose value is 0 and this is known to the seller but not to the buyer. We assume the buyer knows the total number of lemons but not their identity.

For any derivative $F$ on $N$ inputs, input distribution $X$ over $\{0, 1\}^N$, and $n \leq N$, we define the *lemon cost* of $F$ for $n$ junk assets as

$$\begin{aligned}\Delta(n) &= \Delta_{F,X}(n) \\ &= \mathrm{E}[F(X)] \\ &\quad - \min_{S \subseteq [N], |S|=n} E[F(X) | X_i = 0 \forall i \in S]\end{aligned}$$

where the min takes into account all possible ways in which seller could "position" the junk assets among the $N$ assets while designing the derivative. (We'll often drop the subscripts $F, X$ when they are clear from context.) As we discussed earlier, the first term $\mathrm{E}[F(X)]$, which represents the expected value of the derivative when there are no lemons, can be computed approximately with any inversed polynomial precision by Monte Carlo methods. The main difficulty in computing $\Delta(n)$ lies in the minimization in the second term. The lemon cost captures the inefficiency introduced in the market due to the existence of "lemons" or junk assets. When the buyers suspect that there might be $n$ lemons in the assets, he would buy the asset only if the seller is willing to sell him the asset for $\mathrm{E}[F(X)] - \Delta(n)$. (There are economic and regulatory reasons why the seller might be willing to do so, as explained in the companion paper.)

Of course, the above definition of lemon costs ignores the computational complexity of computing $\Delta(n)$. In real life, one must assume that all actors are computationally limited. Accordingly, we will focus on the complexity of computing $\Delta(n)$.

# 3 Worst-case Complexity of Computing Lemon Cost

In this section we show that computing the lemon cost is NP-hard and furthermore even computing approximations is NP-hard.

## 3.1 Derivative Lemon Cost Problem

As we explained in Section 2, a derivative can be viewed as a function $f$ that takes inputs (payoffs of assets or other derivatives), and outputs the payoff. The inputs (payoffs of assets) have a certain known distribution. The derivative pricing problem is to compute the expected value of $f$ given the input distribution. The most common type of functions appear in derivatives are the threshold functions (such as CDSs, CDOs or CDO$^2$s in Section 2). For the input distribution, in many models, the input variables are assumed to be independent. In more complex models, the input variables are assumed to take value according to a *Gaussian copula*, which means they are Gaussians with a block diagonal covariance matrix (the block diagonal covariance matrix models the correlation within a certain area or industry).

Here we consider the difficulty of pricing derivatives when some inputs may be lemons —in other words, the difficulty of evaluating $\Delta(n)$ defined above. In this problem, the seller has $N$ assets (inputs) whose values are i.i.d. distributed (has payoff 0 with probability $p$ and has payoff 1 with probability $1-p$). Up to now there's no difference between this problem and derivative pricing. The key difference is, the seller knows $n$ out of $N$ assets are "lemons" which always gives payoff 0. The buyers only know the number $n$ but not which assets are lemons. So the buyers want to know the lemon cost (as defined in Section 2.1) of the derivatives. We now formally define the class of derivatives we consider and the lemon cost:

**Binary CDO's.**

A (first order) *binary CDO* is a derivative $f$ depending on inputs $x = (x_1, \ldots, x_N)$ that has the following form: $f(x)$ outputs $c$ if $\sum_{j\in S} x_i$ is at least some threshold $t$ and outputs 0 otherwise, where $S \subseteq [N]$, and $c, t \geq 0$. An *order r binary CDO* is a derivative of this form where the $x_i$'s in the sum are replaced with order $(r-1)$ binary CDO's. A *portfolio* of order $r$ binary CDO's, is a derivative $f(x) = f_1(x)+\cdots+f_M(x)$, where each of the $f_i$'s is an order $r$ binary CDO, and for every $x$, $f(x) \leq \sum_{i=1}^{N} x_N$. (This ensures that the seller can always cover the value of the derivative from the underlying assets.) The Derivative Lemon Cost Problem of order $r$ ($DLC[r]$) is defined as the task of, given input a portfolio $f = f_1 + \cdots + f_M$ of order $r$ derivatives on $N$ inputs and a number of lemons $n < N$, computing the number

$$\Delta_{\sum f_i}(n) = \mathrm{E}[\sum_{i=1}^{m} f_i] - \min_{S\subseteq[N],|S|=n} \mathrm{E}[\sum_{i=1}^{m} f_i | x_S = 0]$$

(where $x_S = 0$ means $x_i = 0 \quad \forall i \in S$.)

We show that it's NP-hard to approximate $DLC[1]$ with approximation $1+\epsilon$ for a fixed $\epsilon > 0$, and $DLC[2]$ is NP-hard to approximate up to $\Omega(2^{(logN)^{1/3-\epsilon}})$ for any $\epsilon$. The hardness of approximating a single derivative can be shown by adding an additional level of derivative that computes the sum of $f_i$'s, so our result also implies hardness of approximating the lemon cost of a single derivative of order 2 and 3.

## 3.2 Hardness of $DLC[1]$

**Theorem 1.** *It's NP-hard to approximate $DLC[1]$ with approximation $1+\epsilon$ for a fixed $\epsilon > 0$*

*Proof.* We give a gap-preserving reduction from max independent set with degree bound $d$ to $DLC[2]$. Max independent set with degree bound $d$ is known to be MAX SNP-complete [18]. By their construction we can assume that there are constants $\delta$ and $\epsilon$, such that given a graph $G$ with $k$ vertices and max degree $d$ (think of $d$ as a constant), it's NP-hard to tell whether the graph has an independent set of size $(\delta+\epsilon)k$ or all independent sets of the graph has size smaller than $\delta k$.

From $G$ we construct a graph $G'$, where if a vertex $v \in G$ has degree $d_v < d$, then we add $d - d_v$ vertices in $G'$ and connect them with $v$. The vertices of $G'$ correspond to the inputs, the derivatives are constructed as follows:

For any $e = (u, v) \in G'$, a first order derivative $f_e = u \wedge v$. Let $n = (\delta+\epsilon)k$, it's easy to see the following claim

**Claim 1.1.** *Lemon cost of a particular vertex set in $G'$ is $1/4 \cdot$ (number of edges adjacent to the set).*

So we want to maximize the number of edges adjacent to the vertex set. Notice that the newly added vertices in $G'$ have degree 1 so we'll never choose them, and all original vertices in $G'$ have degree $d$.

When there's an independent set of size $(\delta+\epsilon)k$, we can choose that set and the lemon cost is $(\delta+\epsilon)kd/4$. However, when there's no independent set of size larger than $\delta k$ the lemon cost is at most $(\delta kd + \epsilon k(d-1))/4$. Picking $\epsilon' = \frac{(\delta+\epsilon)d}{\delta d+\epsilon(d-1)} - 1 > 0$, we have thus shown the NP-hardness of computing a $1+\epsilon'$-approximation to $DLC[1]$.

### 3.3 Hardness of $DLC[2]$

We use the hardness of approximating Label Cover to prove the hardness of $DLC[2]$. By results of [19] and [10], we have the following theorem:

**Theorem 2.** *For any constant $\epsilon$, for a regular label cover instance with $k$ vertices and alphabet size $w = \text{poly}(k)$, it's* **NP***-hard to distinguish between the following two cases,*

*Completeness: there is a label assignment that all edges are satisfied.*

*Soundness: no assignment can satisfy more than $2^{-(\log k)^{1-\epsilon}}$ fraction of the edges.*

By reducing Label Cover to $DLC[2]$, we prove that

**Theorem 3.** *For any $\epsilon$, $DLC[2]$ is* **NP***-hard to approximate up to $\Omega(2^{(\log N)^{1/3-\epsilon}})$.*

*Proof.* For a label cover problem with $k$ vertices and alphabet size $w$, we reduce it to a $DLC[2]$ problem with $kw$ assets. When $x_{vl} = 0$, the label $l$ for vertex $v$ is chosen. For each edge $e = (u, v)$ in the label cover, we create a 2nd order derivative $f_e = \wedge_{i=1}^{w}(x_{ui} \vee x_{v\pi(i)})$. Clearly, $f_e = 0$ if and only if the edge is satisfied. We set $p = 1/w$ and $n = k$.

**Completeness**

Choose the set of lemons to be the satisfying assignment. Then all edges are satisfied, so $\sum_e f_e = 0$. Lemon cost is $\mathrm{E}[\sum_{i=1}^{m} f_i] = (1 - 1/w^2)^w m > m/2$.

**Soundness**

With exponentially small probability, the average number of labels will be larger than 3 (the expected value is 2), so the contribution of this case is exponentially small and we ignore that. Let $g(k) = 2^{-(\log k)^{1/3-\epsilon}}$ When average number of labels is at most 3, assume the fraction of edges satisfied is more than $g(k)$. By averaging the fraction of vertices with more than $12/g(k)$ labels chosen is no more than $g(k)/4$, and the fraction of edges that are adjacent to these vertices is at most $g(k)/2$. Thus there are at least $g(k)/2$ fraction of satisfied edges that are adjacent to vertices with less than $12/g(k)$ labels, denote the set of these edges by $S$. For each vertex, if it has labels chosen, randomly pick a label. For edges in $S$, the probability that they are still satisfied is at least $g^2(k)/12^2$, therefore the expected number of satisfied edge in this (partial) assignment is at least $\Omega(g(k)^3)$. However, apply Theorem 2 with $\epsilon' = 4\epsilon$, this is impossible in the soundness case. By contradiction we see the expected number of satisfied assignment is at most $mg(k)$, and lemon cost cannot be larger than that.

Combining the two cases, we showed that it is hard to distinguish whether a $DLC[2]$ instance with $N = kw = poly(k)$ inputs has value at least $m/2$ or $mg(k)$, therefore $DLC[2]$ is hard to approximate up to factor $\Omega(1/g(k)) = \Omega(2^{(\log N)^{1/3-\epsilon}})$ for any $\epsilon > 0$.

## 4 Hardness of Computing Lemon Costs on Average: the Setup

Though the previous section exhibits the worst-case hardness of computing the lemon costs, these hardness results involve special structure in the derivatives and thus have limited implication for real-life markets (this is analogous to the observation about the "factoring derivative" mentioned in the introduction). In this section we try to prove the hardness on more real-life instances, which necessitates, at the minimum, consideration of average-case complexity. In addition, one must exhibit hardness in context of derivatives that "look like" real-life derivatives, and here we will also bring in the notion of *computational indistinguishability* from cryptography.

We start by giving an overview of our construction in Section 4.1, and then explain the notion of lemon cost and cost of complexity in this setting. We introduce the planted dense subgraph and survey the parameter ranges where it is believed to be intractable. The full analysis of lemon costs is fully explained in Section 5.

### 4.1 An Illustrative Example

Consider a seller with $N$ assets (e.g., "mortgages") each of which pays either 0 or 1 with probability 1/2 independently of all others (e.g., payoff is 0 iff the mortgage defaults). Thus a fair price for the entire bundle is $N/2$. Now suppose that the seller has some inside information that an $n$-sized subset $S$ of the assets are actually "junk" or "lemons" and will default (i.e., have zero payoff) with probability 1. In this case the value of the entire bundle will be $(N-n)/2 = N/2-n/2$ and so we say that the "lemon cost" in this setting is $n/2$.

In principle one can use derivatives to significantly ameliorate the lemon cost. In particular consider the following: seller creates $M$ new financial products, each of them depending on $D$ of the underlying assets.[2] Each one of the $M$ products pays off $N/(3M)$

[2]We will have $MD \gg N$ and so the same asset will be contained in more than one product. In modern finance this is normal since different derivatives can reference the same asset. But that one can also think of this overlap between products as occurring from having products that contain assets that are strongly correlated (e.g., mortgages from the same segment of the market). Note that while our model may look simple, similar results can be proven in other models where there are de-

units as long as the number of assets in its pool that defaulted is at most $D/2 + t\sqrt{D}$ for some parameter $t$ (set to be about $\sqrt{\log D}$), and otherwise it pays 0. Henceforth we call such a product a "Binary CDO".[3] Thus, if there are no lemons then the combined value of these $M$ products, denoted $V$, is very close to $N/3$.

One can check (see Section 5) that if the pooling is done randomly (each product depends on $D$ random assets), then even if there are $n$ lemons, the value is still $V - o(n)$, no matter where these lemons are. We see that in this case derivatives do indeed help significantly reduce the lemon cost from $n$ to $o(n)$, thus performing their task of allowing a party to sell off the least information-sensitive portion of the risk.

However, the seller has no incentive to do the pooling completely randomly because he knows $S$, the set of lemons. Some calculations show that his optimum strategy is to pick some $m$ of the CDOs, and make sure that the lemon assets are overrepresented in their pools—to an extent about $\sqrt{D}$, the standard deviation, just enough to skew the probability of default. (Earlier we described this colloquially as "swinging the election.")

Clearly, a fully rational (i.e., computationally unbounded) buyer can enumerate over all possible $n$-sized subsets of $[N]$ and verify that none of them are over-represented, thus computing lemon cost exactly.However, for a real-life buyer who is computationally bounded, this enumeration is infeasible. In fact, the problem of detecting such a tampering is equivalent to the so-called *hidden dense subgraph* problem, which computer scientists believe to be intractable (see discussion below in Section 4.3). Moreover, under seemingly reasonable assumptions, there is a way for the seller to "plant" a set $S$ of such over-represented assets in a way that the resulting pooling will be *computationally indistinguishable* from a random pooling. The bottom line is that under computational assumptions, the lemon cost for polynomial time buyers can be much larger than $n$. Thus introducing derivatives into the picture *amplifies* the lemon cost instead of reducing it!

### 4.2 Cost of Complexity

As we see in the illustrative example, when the binary CDOs are generated by a random distribution, the lemon cost is small, but the seller has the incentive to generate CDOs with another distribution that looks indistinguishable from a random distribution, and the lemon cost in this distribution is much higher than the cost in a random distribution. Because of this computational indistinguishability, we conclude that computationally limited buyers will compute an inaccurate estimate of the lemon cost for at least one of the distributions: they either under-estimate the worth of the truly random derivative, or over-estimate the worth of the derivative with planted dense subgraph, and thus will be off by an amount equal to the difference in the lemon cost for the two derivatives. This difference is the *cost of complexity*. Fully rational buyers will not have this problem because they can distinguish the two distributions and compute the exact value of the lemon cost.

Following this intuition, we define the cost of complexity by analyzing a "pricing game". In the pricing game, Alice gives Bob a contract of derivative $F$ (or derivatives), and Bob needs to compute his estimation of the lemon cost. When Bob has unbounded computational power obviously he can compute the correct value of the lemon cost. However, when Bob is bounded in polynomial time, then it's not surprising that in some situations Bob will not be able to compute the actual lemon cost.

Two distributions over (representations of) derivatives $D_1$ and $D_2$ are $(S, \epsilon)$-*computationally indistinguishable*, denoted by $D_1 \approx_{S,\epsilon} D_2$, if for every circuit $C$ of size at most $S$, $|C(D_1) - C(D_2)| \leq \epsilon$. We define the *cost of complexity* $CoC_{S,\epsilon}$ of a class $\mathcal{F}$ of derivatives (e.g. CDOs on $N$ inputs) and distribution $X$ on the inputs as

$$CoC_{S,\epsilon}(\mathcal{F}, X) = \max_{D_1 \approx_{S,\epsilon} D_2} \mathop{\mathrm{E}}_{D_1}[\Delta_{F,X}(n)] - \mathop{\mathrm{E}}_{D_2}[\Delta_{F,X}(n)] \ .$$

Asymptotically, we define the cost of complexity of an infinite class $\mathcal{F}$ of derivatives to be the minimum cost where $S$ is any arbitrarily large fixed polynomial in the number $N$ of inputs, and $\epsilon > 0$ is any arbitrarily small constant.

**Can the cost of complexity be mitigated?** In Akerloff's classic analysis, the no-trade outcome dictated by lemon costs can be mitigated by appropriate signalling mechanism —e.g., car dealers offering warranties to increase confidence that the car being sold is not a lemon. In the above setting however, there seems to be no direct way for seller to prove that the financial product is *untampered*. (It is believed that there is no simple way to prove the absence of a dense subgraph; this is related to the $NP \neq coNP$ conjecture.). Furthermore, we can show that for suitable parameter choices the tampering is *undetectable* by the buyer even *ex post*. The buyer realizes at the end

pendencies between different assets (e.g., the industry standard Gaussian copula model), see discussion in Section 4.4.

[3]This is a so-called *synthetic binary option*. The more popular collateralized debt obligation (CDO) derivative behaves in a similar way, except that if there are defaults above the threshold (in this case $D/2 + t\sqrt{D}$) then the payoff is not 0 but the defaults are just deducted from the total payoff. We call this a "Tranched CDO". More discussion of binary CDOs appears in Section 4.4.

that the financial products had a higher default rate than expected, but would be unable to prove that this was due to the seller's tampering (see Section 5.1). Nevertheless, we do show in Section 6 that one could use Computer Science ideas in designing derivatives that are tamperproof in our simple setting.

### 4.3 Densest Subgraph Problem

It is convenient to view the relationship of derivatives and assets as a *bipartite graph*, see Figure 1. Derivatives and assets are vertices, with an edge between a derivative and an asset if the derivative depends on this asset.

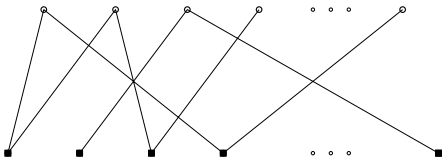

Figure 1: Using a bipartite Graph to represent assets and derivatives. There are $M$ vertices on top corresponding to the derivatives and $N$ vertices at the bottom corresponding to assets. Each derivative references $D$ assets.

Throughout the paper, we'll use the following parameters. We say that an $(M, N, D)$ graph is a bipartite graph with $M$ vertices on one side (which we call the "top" side) and $N$ on the other ("bottom") side, and top degree $D$. We'll often identify the bottom part with assets and top part with derivatives, where each derivative is a function of the $D$ assets it depends on. We say that such a graph $G$ contains an $(m, n, d)$ graph $H$, if one can identify $m$ top vertices and $n$ bottom vertices of $G$ with the vertices of $H$ in a way that all of the edges of $H$ will be present in $G$. We will consider the variant of the *densest subgraph problem*, where one needs to find out whether a given graph $H$ contains some $(m, n, d)$ graph.

Fix $M, N, D, m, n, d$ be some parameters. The (average case, decision) *densest subgraph problem* with these parameters is to distinguish between the following two distributions $\mathcal{R}$ and $\mathcal{D}$ on $(M, N, D)$ graphs:

- $\mathcal{R}$ is obtained by choosing for every top vertex $D$ random neighbors on the bottom.
- $\mathcal{P}$ is obtained by first choosing at random $S \subset [N]$ and $T \subseteq [M]$ with $|S| = n, |T| = m$, and then choosing $D$ random neighbors for every vertex outside of $T$, and $D - d$ random neighbors for every vertex in $T$. We then choose $d$ random additional neighbors in $S$ for every vertex in $T$.

Hardness of this variant of the problem was recently suggested by Applebaum et al [3] as a source for public key cryptography[4]. The state of art algorithms for both the worst-case and average-case problems are from a recent paper of Bhaskara et al [6]. Given their work, the following assumption is consistent with current knowledge:

**Densest subgraph assumption.**

Let $(N, M, D, n, m, d)$ be such that for some positive $\delta$, $N = o(MD)$, $(md^2/n)^2 = o(MD^2/N)$, $n = \Omega(N^{0.5+\delta})$, $m = \Omega(M^{0.5+\delta})$, $d = \tilde{O}(D^{0.5})$ then there is no $\epsilon > 0$ and poly-time algorithm that distinguishes between $\mathcal{R}$ and $\mathcal{P}$ with advantage $\epsilon$.

Since we are not proposing a cryptosystem in this paper, we chose to present the assumption in the (almost) strongest possible form consistent with current knowledge, and see its implications. Needless to say, quantitative improvements in algorithms for this problem will result in corresponding quantitative degradations to our lower bounds on the lemon cost. In this paper we'll always set $d = \tilde{O}(\sqrt{D})$ and set $m$ to be as large as possible while satisfying $(md^2/n)^2 = o(MD^2/N)$, hence we'll have $m = \tilde{O}(n\sqrt{M/N})$.

### 4.4 Our Construction versus Real-life Derivatives

We now discuss on a more technical level the relation of our constructions to commonly used types of derivatives.

**Role of random graph**

Real-life constructions do not use a random graph. Rather, seller tries to put together a diversified portfolio of assets which one can hope are "sufficiently independent." We are modeling this as a random graph. Many experts have worried about the "adverse selection" or "moral hazard" problem inherent because the seller could cherry-pick assets not beneficial to buyer. It is generally hoped that this problem would not be too severe since the seller typically holds on to the junior tranche, which takes the first losses. Unfortunately, this intuition is shown to be false in our model.

**Binary vs tranched.**

Although in most CDOs in practice use "tranching" instead of binary payoff, the results for binary CDOs are still relevant. First, in practice the cost function of a CDO can be quite this continuous, since as soon as the tranche incurs any loss it can suffer from a rating downgrade, and a subsequent significant drop in value. Also, one can take an insurance against such an event, in the form of a *credit default swap (CDS)*,

[4] Applebaum et al used somewhat a different setting of parameters than ours, with smaller planted graphs. We also note that their cryptosystems rely on a second assumption in addition to the hardness of the planted densest subgraph.

hence obtaining a derivative that has approximately the same behavior as the binary CDO we use in this paper. More discussions about tranching will appear in our companion paper.

One significant aspect we ignored in this work is that in standard derivatives the *timing* of defaults makes a big difference, though this seems to make a negative results such as ours only stronger. It is conceivable that such real-life issues could allow one to prove an even better hardness result.

**Asymptotic analysis.**

In this paper we used asymptotic analysis, which seems to work best to bring out the essence of computational issues, but is at odds with economic practice and literature, that use fixed constants. This makes it hard to compare our parameters with currently used ones (which vary widely in practice). We do note that algorithms that we consider asymptotically "efficient" such as semi definite programming, may actually be considered inefficient in current banking environment. Some of results hold also for a threshold of a constant number (e.g. 3) of standard deviations, as opposed to $\sqrt{logD}$ (see Section 5.1), which is perhaps more in line with economic practice in which even the senior most tranche has around 1% probability of incurring a loss (though of course for current parameters $\sqrt{\log D} \leq 3$).

**The role of the model**

The distribution we used in our results, where every asset is either independent or perfectly correlated is of course highly idealized. It is more typical to assume in finance that even for two dissimilar assets, the default probability is slightly correlated. However, this is often modeled via a systemwide component, which is easy to hedge against (e.g. using an industry price index), thus extracting from each asset a random variable that is independent for assets in different industries or markets. In any case, our results can be modified to hold in alternative models such as the Gaussian copula. More discussions involving the model will appear in the companion paper.

## 5 Lemon Cost Bounds for Binary CDOs

In this section we formalize the illustrative example from Section 4.1. We will calculate the lemon cost in "honest" (random) binary CDOs, and the effect on the lemon cost of planting dense subgraphs in such derivatives.

Recall that in the illustrative example there are $N$ assets that are independently and identically distributed with probability 1/2 of a payoff of zero and probability 1/2 of a payoff of 1. In our setting the seller generates $M$ binary CDOs, where the value of each derivative is based on the $D$ of the assets. There is some agreed threshold value $b < B/2$, such that each derivative pays 0 if more than $\frac{D+b}{2}$ of the assets contained in it default, and otherwise pays some fixed amount $V = \frac{D-b}{2D}\frac{N}{M}$ (in the example we used $V = N/(3M)$ but this value is the maximal one so that, assuming each asset participates in the same number of derivatives, the seller can always cover the payment from the underlying assets).

Since each derivative depends on $D$ independent assets, the number of defaulted assets for each derivative is distributed very closely to a Gaussian distribution as $D$ gets larger. In particular, if there are no lemons, every derivative has exactly the same probability of paying off, and this probability (which as $b$ grows becomes very close to 1) is closely approximated by $\Phi(\frac{b}{2\sigma})$ where $\Phi$ is the cumulative distribution function of Gaussian (i.e., $\Phi(a) = \int_{-\infty}^{a} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$), $b$ is our threshold parameter and $\sigma \sim \sqrt{D}$ is the standard deviation. Using linearity of expectation one can compute the expected value of all $M$ derivatives together, which will be about $N\frac{D-b}{2D}\frac{N}{M} \sim N/2$. Note that this calculation is the same regardless of whether the graph is random or not.

We now compute the effect of $n$ lemons (i.e., assets with payoff identical to 0) on the value of all the derivatives. In this case the shape of the pooling *will* make a difference. As in Section 4.3, we use bipartite graph to represent the shape of pooling.

If the seller is honest and pick a random regular graph (we call the distribution of random regular graphs $D_1$), then intuitively the effect of lemons will be small because of law of large numbers.

To increase his expected profit seller can carefully design this graph by choosing a graph with another distribution $D_2$, using his secret information. The key observation is that though each derivative depends upon $D$ assets, in order to substantially affect its payoff probability it suffices to fix about $\sigma \sim \sqrt{D}$ of the underlying assets. More precisely, if $t$ of the assets contained in a derivative are lemons, then the expected number of defaulted assets in it is $\frac{D+t}{2}$, while the standard deviation is $\sqrt{D-t}/2 \approx \sqrt{D}/2$. Hence the probability that this derivative gives 0 return is $\Phi(\frac{t-b}{2\sigma})$ which starts getting larger as $t$ increases. This means that the difference in value between such a pool and a pool without lemons is about $V \cdot \Phi(\frac{t-b}{2\sigma})$.

Suppose the seller allocates $t_i$ of the junk assets to the $i$th derivative. Since each of the $n$ junk assets are contained in $MD/N$ derivatives, we have $\sum_{i=1}^{M} t_i =$

$\frac{nMD}{N}$. In this case the lemon cost will be

$$V \cdot \sum_{i=1}^{M} \Phi(\frac{t_i - b}{2\sigma})$$

Since the function $\Phi(\frac{t_i-b}{2\sigma})$ is concave when $t < b$, and convex after that the optimum solution will involve all $t_i$-s to be either 0 or $k\sqrt{D}$ for some small constant $k$. (There is no closed form for $k$ but it is easily calculated numerically; see Appendix A.)

Therefore the lemon cost is maximized by choosing some $m$ derivatives and letting each of them have at least $d = k\sqrt{D}$ edges from the set of junk assets. In the bipartite graph representation, this corresponds to a dense subgraph, which is a set of derivatives (the manipulated derivatives) and a set of assets (the junk assets) that have more edges between them than expected. This precisely corresponds to the pooling graph containing an $(m, n, d)$ subgraph - that is a *dense subgraph* (we sometimes call such a subgraph a "booby trap"). When the parameters $m$, $n$, $d$ are chosen carefully, there will be no such dense subgraphs in random graphs with high probability, and so the buyer will be able to verify that this is the case. On the other hand, assuming the intractability of this problem, the seller will be able to embed a significant dense subgraph in the pooling, thus significantly raising the lemon cost. We call the distribution of derivatives with planted dense subgraph $D_2$.

Note that even random graphs have dense subgraphs. For example, when $md = n$, any graph has an $(m, n, d)$ subgraph— just take any $m$ top vertices and their $n = md$ neighbors. But these are more or less the densest subgraphs in random graphs, as is shown by the following proposition, that is proven via a standard combination of Chernoff bound and the union bound:

**Proposition 3.1.** *If $m(d-1) > n+\omega(1)$, and $\frac{dN}{Dn} > (N+M)^{\epsilon}$ for some constant $\epsilon$, then there is no dense subgraph $(m, n, d)$ in a random $(M, N, D)$ graph with high probability.*

The above discussion allows us to quantify precisely the effect of an $(m, n, d)$-subgraph on the lemon cost. Let $p \sim \Phi(-b/2\sigma)$ be the probability of default. The mere addition of $n$ lemons (regardless of dense subgraphs) will reduce the value by about $\Phi'(-b/2\sigma)\frac{nD}{2N}\frac{1}{\sigma} \cdot N/2 = O(e^{-(b/2\sigma)^2/2} n\sqrt{D})$ which can be made $o(n)$ by setting $b$ to be some constant time $\sqrt{D \log D}$. The effect of an $(m, n, d)$ subgraph on the lemon cost is captured by the following theorem:

**Theorem 4.** *When $d - b > 3\sqrt{D}$, $n/N \ll d/D$, an $(m, n, d)$ subgraph will generate an extra lemon cost of at least $(1 - 2p - o(1))mV$.*

*Proof.* For each manipulated derivatives, let $Y$ be the number of defaulted assets, since there are $d$ assets that come from junk asset set, these $d$ assets will always default, and the expectation of $Y$ is $\mathrm{E}[Y] = \frac{D+d}{2}$. Since $D$ is large enough so that the Gaussian approximation holds, $\Pr[Y \geq \frac{D+b}{2}] = 1 - p$.

For non-manipulated derivatives, assume the expected number of defaulted assets is $x$, then $x$ satisfy the equation:

$$m \cdot \frac{D+d}{2} + (M-m) \cdot x = \frac{N+n}{2N} \cdot \frac{MD}{N},$$

because the LHS and RHS are all expected number of defaulted mini-assets. $x \geq \frac{D}{2} + \frac{n}{2N} - \frac{md}{2M-2m}$. The probability that the number of defaulted assets is more than $\frac{D+b}{2}$ is at least $\Phi(-3 - \frac{md}{2(M-m)D}) = p - \Phi'(-3)\frac{md}{2(M-m)D} = p - O(\frac{md}{2(M-m)D})$. The expected number of non-manipulated derivatives that gives no return is at least $p(M-m) - O(\frac{md}{2D}) = p(M-m) - o(m)$.

In conclusion, the expected number of derivatives that gives no return is at least $pM + (1-2p)m - o(m)$. which is $(1 - 2p - o(1))$ smaller than the expectation without dense subgraphs. Thus the extra lemon cost is $(1 - 2p - o(1))mV$.

Since in $D_1$ (random graph) there's no large dense subgraph, while in $D_2$ there is a planted dense subgraph, using this quantification of the effect of dense subgraph, we show a lower bound on the cost of complexity for binary CDO's:

**Theorem 5.** *Assuming the densest subgraph assumption, then there is some $\epsilon > 0$ such that for every $\delta > 0$ if $N = M^{1+\delta}$ then the cost of complexity for $n$ ($n = \omega(\sqrt{N}M^{\delta})$ lemons and the family of portfolios of $M$ binary first order binary CDO's over $N$ inputs is at least $\tilde{\Omega}(n\sqrt{N/M})$.*

*Proof.* Let $D = M^{2\delta}$, $d = \tilde{\Theta}(\sqrt{D})$, $m = \tilde{\Theta}(n/M^{\delta})$, then the parameters satisfy the requirements of densest subgraph assumption, the distributions $D_1$ (random graph) and $D_2$ (random graph with planted $(m, n, d)$-graph) is computationally indistinguishable.

For distribution $D_1$, by Proposition 3.1 there is no $(N^{\epsilon}n/d, n, d)$ dense subgraph for any $\epsilon$. Therefore by Theorem 4, the extra lemon cost is at most $N^{\epsilon}n/dV = O(n\sqrt{D}M/N^{1-\epsilon})$.

For distribution $D_2$, by Theorem 4, the extra lemon cost is at least $\Omega(mV) = \tilde{\Omega}(n\sqrt{N/M}) \gg O(n\sqrt{D}M/N^{1-\epsilon})$

Therefore,

$$|\operatorname*{E}_{D_1}[\Delta(n)] - \operatorname*{E}_{D_2}[\Delta(n)]| = \tilde{\Omega}(n\sqrt{N/M}),$$

and by definition, $CoC = \tilde{\Omega}(n\sqrt{N/M})$.

### 5.1 Ex-Post Detection: Lemon Laws May not Help

Our construction above was such that it is difficult to detect the presence of dense subgraphs (i.e., booby traps) at the time of sale of the derivative, before it is known which assets pay off and which ones default (i.e., *ex ante*). This leads to the question whether buyers could try to enforce honest construction of the derivative by including a clause in the contract that says for example that that the derivative designer pays a huge fine if a dense subgraph (denser than should exist in a random construction) is found in the derivatives vs. assets bipartite graphs. Since the buyer can wait to detect this subgraph *after* all assets results are known we call this *ex post detection* of dense subgraphs[5]. Alternatively, the government may pass a "lemon law" for derivatives that protects buyers. In this section we will show that these actions may not help much.

First, note that Ex-Post detection is potentially easier than Ex-Ante detection, because the buyer has more information after the result has been revealed: the junk assets must be assets in the set of the defaulted assets, and most of the manipulated derivatives are in the set of derivatives with no return. One simple use of this observation is: when the expected number of derivatives with no return in random graphs is extremely small (e.g. $o(m)$), then almost all derivatives with no return are in the dense subgraph, and hence one can find it easily. But the number of derivatives with no return is large (e.g. $\Omega(M)$) then it still seems hard to find the dense subgraph even ex-post. We do not have a proof that this is the case, but can show a relation between finding the planted subgraph in the ex-post and ex-ante setting in a slightly differe nt model of random graphs.

Fixing parameters $N, M, D, n, m, d$ as usual we define the *added dense subgraph search problem* to be the following problem: one is given a bipartite graph $G$ on $M+N$ vertices that is constructed by **(a)** having each vertex $v_i$ of the $M$ top vertices of $G$ choose a number $D_i$ using a geometric random variable with expectation $D$, and then choose $D_i$ random neighbors, and **(b)** adding to $G$ the edges of a random bipartite graph $H$ on $m+n$ vertices and degree $d$, where we place $H$ on $m$ random top vertices and $n$ random bottom vertices. The goal in ex poste detection is to recover the vertices of $H$ or any other induced subgraph of $G$ on $m+n$ vertices that has $\omega(n)$ edges.

Since we'll be interested in $d = \Omega(\sqrt{D})$ and $m = \Omega(\sqrt{M})$, it will be in fact easy to use the degree distribution of such a graph $G$ to distinguish it from a random graph in which step **(b)** is not applied. But the point is that it might still be hard to actually find the dense subgraph. If this is hard, then we show it's still hard even in the *ex-post* setting.

**Theorem 6.** *Suppose that the added dense subgraph search problem is hard for parameters $N, M, D, n, m, d$ then the search problem is still hard for parameters $2N, M, 2D, n, m, d$ given a random assignment $x \in \{0,1\}^N$ conditioned on $x_i = 0$ for all $i$ in the planted graph.*

*Proof sketch.* Given an instance $G$ of the dense subgraph we'll convert it into an instance $(G', x)$ of the ex-pose problem, where $G'$ is a graph on $M+2N$ vertices and degree parameter roughly $2D$, and $x$ is an assignment to the $2N$ bottom vertices of $G'$ on which all vertices in the planted graph are zero.

The graph $G$ will contain the $M+N$ vertices of $G$ but we add to it $N/2$ new bottom nodes, and let $x$ be an assignment that gives 0 to all old nodes and 1 to all new ones. (We'll later permute the nodes randomly; also in this proof sketch we assume that the assignment is balanced, but in fact we'll need to choose the number of nodes to add in a way that the number of 0's of the resulting assignment is distributed according to the binomial distribution.)

Given such a balanced assignment $x$ to the bottom vertices, adding edges to a top vertex $v$ in this model to a graph can be thought of as the following process: one tosses a 3-way coin that with probability $1/D$ outputs "halt", with probability $(1-1/D)/2$ outputs 0 and with probability $(1-1/D)/2$ outputs 1. If the coin outputs 0 we add to $v$ a random neighbor with 0 assignment, if it outputs 1 we add a random neighbor with a 1 assignment, and continue until the coin says "halt".

[5] We remark that this is not the only, nor necessarily the best, way to mitigate these problems. Another could be for the designer to use random coins that were generated by some public trusted authority. The problem in both these approaches is that sometimes the graph is constructed by the seller subject to a variety of constraints, and hence the seller himself may not know for sure if a dense subgraph exist. Now if a dense subgraph exists without the sellers knowledge, this opens the possibility that the buyer will find it, either by luck or by investing significant computational effort, and hence use a clause such as above to extract the fine from the seller. Note that making the fine equal to the derivative's value is not a solution as well, since that allows the party that knows the existence of the dense subgraph to decide based on its knowledge of the past to decide if it wants to "roll back the deal".

We now imitate this process but starting with the edges already in $G$: for each top vertex $v$ in $G'$, we conceptually toss such a three way coin, but add an edge only if it outputs 1: for the 0 or "halt" possibilities we defer to the coin tosses made in the generation of $G$. More concretely, we set $\epsilon \sim 1/(2D)$ to satisfy the equation $1/2 - \epsilon = (1/2 + \epsilon)(1 - 1/D)$. We now toss a $1/2 - \epsilon$ biased coin and if it comes out "heads" we add to $v$ a random 1 neighbor (e.g. one of the new $N$ vertices we added), if it comes out "tails" we do nothing. We continue to do so until the coin comes out "tails" $D_i + 1$ times, where $D_i$ is the degree of $v$ in $G$.

Because this model does not fit with our standard model of planted graphs (and indeed cannot, since we need a model where not only the *search* but also the *detection* problem can be hard), we do not have any formal results for this model. However, many of our results hold even when the threshold is set to a constant number of standard deviations. In this case it seems plausible to conjecture that the ex-post search problem still remains hard as well.

# 6 Design of Derivatives Resistant to Tampering

The results of this paper show how the usual methods of CDO design are susceptible to manipulation by sellers who have hidden information. This raises the question whether some other way of CDO design is less susceptible to this manipulation. This section contains a *positive* result, showing more exotic derivatives that are not susceptible to the same kind of manipulation. Note that this positive result is in the simplified setup used in the rest of the paper and it remains to be seen how to adapt these ideas to more realistic scenarios with more complicated input correlations, timing assumptions, etc.

The reason current constructions (e.g., the one in our illustrative example) allow tampering is that the financial product is defined using the sum of $D$ inputs. If these inputs are iid variables then the sum is approximately gaussian with standard deviation $\sqrt{D}$, and thus to shift the distribution it suffices to fix only about $\sqrt{D}$ variables. This is too small for buyers (specifically, the best algorithms known) to detect.

The more exotic derivatives proposed here will use different functions, which cannot be shifted without fixing a lot of variables. This means that denser graphs will be needed to influence them, and such graphs could be dense enough for the best algorithms to find them.

## 6.1 XOR instead of Threshold

As a warmup we give the more exotic of the two constructions, since its correctness is easier to show. Instead of simple sum of the variables, we use $XOR$ or sum modulo 2.

**Theorem 7.** *Suppose in our illustrative example we use XOR instead of simple sum, so that the product produces a payoff iff the XOR of the underlying variables is* 1. *When $D \gg \log M + \log 1/\epsilon$, a dense subgraph that can change the expectation of even a single particular output (derivative) by $\epsilon$ can be detected Ex-Post.*

*Proof.* According to the construction, all assets outside the dense subgraph are independent and have probability $\rho$ of being 1. (In our illustrative example, $\rho = 1/2$ but any other constant will work below as well.) If the degree of the dense subgraph $d = D - t$, then by property of $XOR$ function, the expectation of the $XOR$ function lies in $1/2 \pm c^t$ where $c$ is a constant smaller than 1 that only depends on $\rho$. When $t = c' \log 1/\epsilon$, $c^t$ is smaller than $\epsilon$. Therefore $d > D - O(\log 1/\epsilon)$. The probability that $d$ of the inputs are all 0 is exponentially small in $D$, and thus is smaller than $1/M$. That is, the expected number of derivative with at least $d$ inputs are 0 is less than 1. But all derivatives in the dense subgraph satisfy this property. These derivatives are easily detected in the Ex-Post setting.

The $XOR$ function is even effective in preventing dense subgraphs against a more powerful adversary and Ex-Ante setting, where the adversary does not only know that there are $n$ junk assets, he can actually control the value of these $n$ assets.

**Theorem 8.** *When $D \gg \log 1/\epsilon$, dense subgraphs are Ex-Ante detectable if the ratio of manipulated output $(m/M)$ is larger than the ratio of manipulated input $n/N$.*

*Proof.* Similarly, we have $d > D - O(\log 1/\epsilon) = D - o(D)$. If $m/M > n/N$, and $MD \geq N$, then

$$\frac{m^4 d^8}{n^4} \geq \frac{M^4 d^8}{N^4} = \Theta(\frac{M^4 D^8}{N^4}) \gg \frac{M^3 D^7}{N^4}$$

so whether there is a dense subgraph can be detected by cycle counting algorithm (see Section 7.2)

## 6.2 Tree of Majorities

The $XOR$ function is not too natural in real life. One particular disadvantage is that $XOR$ function is

not *monotone*. Buyers would probably insist upon the property that the derivative should be rise and fall with the number of assets that default, which is not true of $XOR$. Now we give a different function that is monotone but which still deters manipulation.

Let us think through the criteria that such a function must satisfy. Of course we would want the function to be more resistant to fixing a small portion of its inputs. And, the function needs to have symmetry, so that each input is treated the same as all other inputs. (We suspect that if the function is not symmetric then it makes manipulation easier.) Finally, we want the function to be *securitizable* and give a large payoff, that is, when the value of the function is 1, the sum of inputs should exceeds a certain (large) threshold, so the seller can use the payoffs of the inputs to pay for the derivative. A standard function that is monotone, symmetric and not sensitive to fixing a small portion of input is the tree of majorities (appeared in [4], [1] gave a function that is more resistant but is randomly constructed and not as "natural" as tree of majorities in economics):

**Definition 1** (Tree of Majority)**.** The function $TM^k$ on $3^k$ inputs is defined recursively:

$$TM^0(x) = x$$

$$\begin{aligned} TM^k(x_1, x_2, ..., x_{3^k}) =& MAJ(TM^{k-1}(x_1, x_2, ..., x_{3^{k-1}}), \\ & TM^{k-1}(x_{3^{k-1}+1}, ..., x_{2\cdot 3^{k-1}}), \\ & TM^{k-1}(x_{2\cdot 3^{k-1}+1}, ..., x_{3^k})) \end{aligned}$$

In this section we assume all "normal" assets are independent and will default with probability 1/2. It's clear that $TM^k$ is 1 with probability 1/2, is monotone and has some symmetry properties. This function is also resistant to fixing $o(2^k)$ inputs, as is shown in the following theorem:

**Theorem 9.** *Fixing any $t$ inputs to 0, while other inputs are independent 0/1 variable with probability 1/2 of being 1, the probability that $TM^k$ is 0 is at most $(1 + t/2^k)/2$.*

*Proof.* By induction.

When $k = 0$ the theorem is trivial.

Assume the theorem is true for $k = n$, when $k = n + 1$, since $t$ inputs are fixed, let the number of fixed inputs in the 3 instances of $TM^n$ be $t_1$, $t_2$, $t_3$ respectively. Then $t_1 + t_2 + t_3 = t$. When $t \geq 2^{n+1}$ the theorem is trivially true. When any of $t_1$, $t_2$, $t_3$ is more than $2^n$, then it's possible to reduce it to $2^n$ while still make sure that instance of $TM^n$ is always 1. Now, the $i$-th instances of $TM^n$ have at most $(1+t_i/2^n)/2$ probability of being 0, so the probability that $TM^{n+1}$ is 0 is at most

$$\begin{aligned} & \frac{1+\frac{t_1}{2^n}}{2}\frac{1+\frac{t_2}{2^n}}{2}\frac{1+\frac{t_3}{2^n}}{2} + \frac{1-\frac{t_1}{2^n}}{2}\frac{1+\frac{t_2}{2^n}}{2}\frac{1+\frac{t_3}{2^n}}{2} \\ & + \frac{1+\frac{t_1}{2^n}}{2}\frac{1-\frac{t_2}{2^n}}{2}\frac{1+\frac{t_3}{2^n}}{2} + \frac{1+\frac{t_1}{2^n}}{2}\frac{1+\frac{t_2}{2^n}}{2}\frac{1-\frac{t_3}{2^n}}{2} \\ =& \frac{1}{2} + \frac{1}{4}\left(\frac{t_1}{2^n} + \frac{t_2}{2^n} + \frac{t_3}{2^n}\right) - \frac{1}{4}\frac{t_1}{2^n}\frac{t_2}{2^n}\frac{t_3}{2^n} \\ \leq& \frac{1+\frac{t}{2^{n+1}}}{2} \end{aligned}$$

**Theorem 10.** *Suppose in the illustrative example the financial products pay 1 iff the tree function $TM^k$ (where $D = 3^k$) of the $D$ underlying mini-assets evaluates to 1. If $D \gg \log^3 M$ then any manipulation that significantly changes the expectation of even a single financial product is ex-post detectable.*

*Proof.* By the above analysis, to control the output value of $TM$ function, the dense subgraph parameter needs to satisfy $d = \Omega(D^{2/3})$. Thus when $D \gg \log^3 M$ this is ex-post detectable because the likelihood of $d$ inputs being 0 in some financial product is so small that every such product must come from the dense subgraph.

The function $TM^k$ itself is not *securitizable*, because when the value of the function is 1, we are only guaranteed that $2^k$ of the inputs are 1, and will be able to pay $2^k$, while the total value can be as large as $O(3^k)$. An easy way to fix this is to take $F = TM^k \wedge Threshold(0.4 \cdot 3^k)$, since $Threshold(0.4 \cdot 3^k)$ is almost always 1 in the given distribution, the function $F$ acts the same as $TM^k$, but when $F = 1$, the number of ones in the input is at least $0.4 \cdot 3^k$, and we would be able to pay using the value of these assets.

## 7 Detecting Dense Subgraphs: Survey

Since designers of financial derivatives can use *dense subgraphs* in their products to benefit from hidden information, it would be useful for buyers and rating agencies to actually search for such anomalies in financial products. Currently rating agencies seem to use simple agorithms based upon monte carlo simulation [20]. There is some literature on how to design derivatives, specifically, using ideas such as *combinatorial designs* [13]. However, these design criteria seem too weak.

Now we survey some other algorithms for detecting dense subgraphs, which are more or less straightforward using known algorithm design techniques. The goal is to quantify the range of dense subgraph parameters that will make the dense subgraph *are* detectable

using efficient algorithms. The dense subgraph parameters used in our constructions fall outside this range.

Note that the efficient algorithms given in this section require full access to the entire set of financial products sold by the seller, and thus assumes a level of transparency that currently does not exist (or would be exist only in a law enforcement setting). Absent such transparency the detection of dense subgraphs is much harder and the seller's profit from planting dense subgraphs would increase.

In this section we'll first describe several algorithms that can be used to detect dense subgraphs; then we introduce the notion of Ex-Post detection; finally we discuss the constraints the booby trap needs to satisfy.

### 7.1 Co-Degree Distribution

We can expect that some statistical features may have a large difference that allow us to detect this dense subgraph. The simplest statistical features are degree and co-degree of vertices. The degree of a vertex $d(v)$ is the number of edges that are adjacent to that vertex. However, in our example, we make sure that each asset is related to the same number of derivatives and each derivative is related to the same number of assets. Therefore in both the random graph and random graph with planted dense subgraph, the degrees of vertices appear the same. The co-degree of two vertices $u$ and $v$, $cod(u, v)$ is the number of common neighbors of $u$ and $v$. When $\frac{D^2}{N} \gg 1$, by the law of large numbers, the co-degree of any two derivatives is approximately a Gaussian distribution with mean $D^2/N$ and standard deviation $\sqrt{D^2/N}$. In the dense subgraph, two derivatives are expected to have $d^2/n$ more common neighbors than a pair of vertices outside the dense subgraph. When

$$\frac{d^2}{n} > \sqrt{\frac{D^2 \log N}{N}},$$

by property of Gaussian distribution we know that with high probability, the pairs with highest co-degree will all be the pairs of vertices from the dense subgraph. Therefore by computing the co-degree of every pair of derivatives and take the largest ones we can find the dense subgraph.

### 7.2 Cycle Counting

Besides degree and co-degree, one of the features we can use is the number of appearance of certain "structure" in the graph. For example, the number of length $2k$ cycles, where $k$ is a constant, can be used in distinguishing graph with dense subgraph. In particular, counting the number of length 4 cycles can distinguish between graphs with dense subgraph or truly random graphs in some parameters.

For simplicity we assume the following distributions for random graphs. The distribution $\hat{\mathcal{R}}$ is a random bipartite graph with $M$ top vertices and $N$ bottom vertices and each edge is chosen independently with probability $D/N$. The distribution $\hat{\mathcal{D}}$ is a random bipartite graph with $M$ top vertices and $N$ bottom vertices, there is a random subset $S$ of bottom vertices and a random subset $T$ of top vertices, $|S| = n$ and $|T| = m$. Edges incident with vertices outside $T$ are chosen independently with probability $D/N$. Edges between $S$ and $T$ are chosen with probability $d/n$. Edges incident to $T$ but not $S$ are chosen with probability $D - d/(N - n)$.

Let $\alpha$ be a length 4 cycle, then $X_\alpha$ is the indicator variable for this cycle (that is, $X_\alpha = 1$ if the cycle exists and 0 otherwise). Throughout the computation we will use the approximation $M - c \sim M$ where $c$ is a small constant. Let $R$ be the number of length 4 cycles in $\hat{\mathcal{R}}$, then $R = \sum_\alpha X_\alpha$.

$$\mathrm{E}[R] = \sum_\alpha \mathrm{E}[X_\alpha] = \frac{M^2}{2}\frac{N^2}{2}\left(\frac{D}{N}\right)^4 = \frac{M^2 D^4}{4N^2}$$

The variance of $R$ is

$$\mathrm{Var}[R] = \mathrm{E}[R^2] - \mathrm{E}[R]^2 = \sum_{\alpha,\beta}(\mathrm{E}[X_\alpha X_\beta] - (D/N)^8)$$

Since each $\mathrm{E}[X_\alpha X_\beta]$ is at least $(D/N)^8$, this is lower bounded by $\sum_{(\alpha,\beta)\in H}(\mathrm{E}[X_\alpha X_\beta]-(D/N)^8)$ for any set $H$. It's easy to check that the dominating term comes from the set $H$ where $\alpha$ and $\beta$ shares a single edge. $|H| = M^3N^3/4$, and for each $(\alpha, \beta) \in H$, $\mathrm{E}[X_\alpha X_\beta] = (D/N)^7$. Therefore $\mathrm{Var}[R] = \Omega(M^3D^7/N^4)$. By considering cycles that share more edges we can see that the terms they introduce to $\mathrm{Var}[R]$ are all smaller, so $\mathrm{Var}[R] = \Theta(M^3D^7/N^4)$.

For $\hat{\mathcal{D}}$, when the two top vertices of $\alpha$ are all outside $T$ $\mathrm{E}[X_\alpha] = (D/N)^4$; when one of the vertices is inside $T$ the average value for $\mathrm{E}[X_\alpha]$ is still $(D/N)^4$ (this can be checked by enumerating the two bottom vertices); when two top vertices are all in $T$ the average value for $\mathrm{E}[X_\alpha]$ is roughly $d^4/n^2N^2$. Let $Y$ be the number of 4-cycles in $\hat{\mathcal{D}}$, then

$$\mathrm{E}[Y] - \mathrm{E}[R] \approx \frac{m^2}{2}\frac{N^2}{2}\frac{d^4}{n^2N^2} = \frac{m^2d^4}{4n^2}$$

That is, when $m^4d^8/n^4 \gg \mathrm{Var}[R] = \Theta(M^3D^7/N^4)$, by counting length 4 cycles the two distributions $\hat{\mathcal{R}}$ and $\hat{\mathcal{D}}$ can be distinguished.

### 7.3 Semi-definite Programming

Semi-definite programming (SDP) is a powerful tool to approximate quantities we do not know how to compute. Consider the densest bipartite subgraph problem: given a bipartite graph with $N$ vertices on the left side and $M$ vertices on the right side, find a set of $n$ vertices from left side and $m$ vertices on the right side, such that the number of edges between them is maximized. Clearly if we can solve this problem we would be able to see whether there's a dense subgraph in the graph. Because the dense subgraph is much denser than average, we can distinguish a random graph with one with dense subgraph even if we can approximate the densest subgraph problem to some ratio.

Using SDP, we can compute a value $v_{SDP}$ that is no smaller than the number of edges in the densest subgraph. Using a simple extension of the SDP from [6], for random graph, $v_{SDP}$ is of order $\Theta(\sqrt{Dmn})$. If the number of edges in the dense subgraph ($md$) is larger than this (which means $m \gg n$), then since the SDP value of the graph with dense subgraph is at least $md$, the SDP algorithm can be used to distinguish the two situations.

### 7.4 Log Density

In [6] Charikar et al. purposed a new algorithm to detect dense subgraphs. A central concept in their algorithm is *log*-density. The *log*-density of a graph is the ratio between logarithm of average degree and the logarithm of number of vertices. When the planted dense subgraph has higher *log*-density than the whole graph, their algorithm can detect the dense subgraph in polynomial time. Although their algorithm does not directly extend to the bipartite case, for safety we assume that $\log d/\log m < \log D/\log M$ and $\log d/\log n < \log D/\log N$.

## 8 Conclusions

This paper has argued that computational complexity has a role to play in understanding finance, because (a) sheer number of assets, financial products etc. makes asymptotic analysis relevant (b) economic actors have an incentive to make use of this sheer amount of information to *hide* their self-interested actions.

Most analysis of the current financial crisis blames the use of "faulty models" in pricing derivatives rather than computational complexity and we do not dispute that effect. Coval et al. [8] give a readable account of this "modelling error", and point out that buyer practices (and in particular the algorithms used by rating agencies [20]) do not involve bounding the lemon cost or doing any kind of sensitivity analysis of the derivative other than running Monte-Carlo simulations on a few industry-standard distributions such as the Gaussian copula. (See also [7].)

However, this raises the question whether a more precise model would insulate the market from future problems. Our results can be seen as some cause of concern, even though they are clearly only a first step (simple model, asymptotic analysis, etc.). The lemon problem clearly exists in real life (e.g., "no documentation mortgages"), and there will always be a discrepancy between the buyer's "model" of the assets and the true valuation. Since we exhibit the computational intractability of pricing even when the input model is known ($N-n$ independent assets and $n$ junk assets), one fears that such pricing problems will not go away even with better models. If anything, the pricing problem should only get harder for more complicated models. (Our few positive results in Section 6 raise the hope that it may be possible to circumvent at least the tampering problem with better design.) In any case, we feel that from now on computational complexity should be explicitly accounted for in the design and trade of derivatives.

Several questions suggest themselves.

1. Is it possible to prove even *stronger* negative results, either in terms of the underlying hard problem, or the quantitative estimate of the lemon cost? In our model, solving some version of densest subgraph is necessary and sufficient for detecting tampering. But possibly by considering more lifelike features such as *timing* conditions on mortgage payments, or more complex input distributions, one can embed an even more well-known hard problem. Similarly, it is possible that the lemon costs for say tranched CDOs are higher than we have estimated.
2. Is it possible to give classes of derivatives (say, by identifying suitable parameter choices) where the cost of complexity goes away, including in more lifelike scenarios? This would probably involve a full characterization of all possible tamperings of the derivative, and showing that they can be detected.
3. If the previous quest proves difficult, try to prove that it is impossible. This would involve an axiomatic characterization of the goals of securitization, and showing that no derivative meets those goals in a way that is tamper-proof.

## Acknowledgments

We thank Moses Charikar for sharing with us the results from the manuscript [6].

## References


[1] M. Ajtai and N. Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993.

[2] G. Akerlof. The market for "lemons": quality uncertainty and the market mechanism. *The quarterly journal of economics*, pages 488–500, 1970.

[3] B. Applebaum, B. Barak, and A. Wigderson. Public key cryptography from different assumptions. Available from the authors' web pages. Preliminary version as cryptology eprint report 2008/335 by Barak and Wigderson., 2009.

[4] M. Ben-Or and N. Linial. Collective coin flipping. *Randomness and Computation(S. Micali ed.)*, pages 91–115, 1990.

[5] A. Bernardo and B. Cornell. The valuation of complex derivatives by major investment firms: Empirical evidence. *Journal of Finance*, pages 785–798, 1997.

[6] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, and A. Vijayaraghavan. Detecting high log-density: An $o(n^{1/4})$-approximation for densest k-subgraph. Manuscript in preparation, 2009.

[7] M. Brunnermeier. Deciphering the 2007-08 liquidity and credit crunch. *Journal of Economic Perspectives*, 23(1):77–100, 2008.

[8] J. Coval, J. Jurek, and E. Stafford. The economics of structured finance. *Journal of Economic Perspectives*, 23(1):3–25, 2009.

[9] P. DeMarzo. The pooling and tranching of securities: A model of informed intermediation. *Review of Financial Studies*, 18(1):1–35, 2005.

[10] I. Dinur and S. Safra. On the hardness of approximating label cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 6(15), 1999.

[11] D. Duffie. Innovations in credit risk transfer: implications for financial stability. *BIS Working Papers*, 2007.

[12] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[13] P. Flener, J. Pearson, L. G. Reyna, and O. Sivertsson. Design of financial cdo squared transactions using constraint programming. *Constraints*, 12(2):179–205, 2007.

[14] G. Gigerenzer, R. Selten, and eds. *Bounded rationality : the adaptive toolbox*. MIT Press, 2002.

[15] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *FOCS*, pages 136–145, 2004.

[16] R. McDonald. *Derivatives markets*. Addison-Wesley Reading, MA, 2003.

[17] C. Mounfield. *Synthetic CDOs: Modelling, Valuation and Risk Management*. Cambridge Univ Pr, 2008.

[18] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

[19] R. Raz. A parallel repetition theorem. In *STOC*, pages 447–456, 1995.

[20] M. Whelton and M. Adelson. *CDOs-squared demystified*. Nomura Securities, February 2005.


## A Maximizing Profit in Binary CDO Setting

We provide a more complete analysis than the one in Section 5 of the full optimization problem for the seller. Recall that his profit is given by $V' \cdot \sum_{i=1}^{M} \Phi(\frac{t_i - b}{2\delta})$, yet at the same time he is not allowed to put too many edges into the cluster (otherwise this dense subgraph can be detected). We abstract this as an optimization problem (let $\frac{b}{2\delta} = a$, $x_i = t_i/2\delta$):

$$\begin{aligned} \max \quad & \sum_{i=1}^{M} \Phi(x_i - a) \\ \text{subject to} \quad & \sum_{i=1}^{M} x_i = C \\ & x_i \geq 0 \end{aligned}$$

As argued in Section 5, each $x_i$ should either be 0 or greater than $a$, so it's safe to subtract $M\Phi(-a)$ from the objective function. Now the sum of $x_i$'s are fixed, and we want to maximize the sum of $\Phi(x_i - a) - \Phi(-a)$, clearly the best thing to do is to maximize the ratio between $\Phi(x_i - a) - \Phi(-a)$ and $x_i$. Let $y = \Phi(x-a) - \Phi(-a)$, the $x$ that maximize $y/x$ is the point that satisfies $y = x \cdot y'(x)$, that is

$$\Phi(x - a) - \Phi(-a) = x \frac{e^{-\frac{(x-a)^2}{2}}}{\sqrt{2\pi}},$$

when $x \geq a + 3 + \sqrt{4 \log a}$, $LHS \approx 1$, $RHS < 1$ (when $a$ is large this is bounded by $\sqrt{4 \log a}$ term, when $a$ is small this is bounded by 3, the constants are not carefully chosen but they are large enough), so the solution cannot be in this range. When $a$ is a constant, the solution $x$ lies between $a$ and $a+3+\sqrt{4 \log a}$. Therefore, when $a$ us a constant, the best thing to do is to set all non-zero $x_i$'s to a value not much larger than $a$, in the bipartite graph representation this is just planting a dense subgraph where each derivatives have $O(\sqrt{D})$ edges.

## B Constraints for Parameters

In order to avoid detection and satisfy the requirement of making profit, the parameters for dense subgraph need to be carefully chosen. Here we give a list of constraints that need to be satisfied by the parameters in different models.

1. $MD \geq N$.
   This is a trivial constraint saying that each asset should be related to at least one derivative.
2. $d - b > 3\sqrt{D}$.
   This constraint makes sure that the derivatives in the dense subgraph behaves differently from the derivatives outside. When $d - b > \sqrt{D}$, by Gaussian approximation, the probability that the number of defaulted assets is more than $\frac{D+b}{2}$ is at least $\Phi(3) = 99.8\%$. When $d-b$ is smaller, say $d-b = \sqrt{D}$, the probability of the same event will be much smaller. Especially when $d-b = o(\sqrt{D})$ such event will have a subconstant probability.
3. $d/D \gg n/N$.
   This constraint says that the dense subgraph is much denser than average. If it is not satisfied, then any derivative is expected to contain $D \cdot n/N > d$ assets in the junk asset set, even simple monte-carlo simulation will be able to see that the expected payoff value of the derivatives has changed because of the junk assets.
4. $n \ll md$.
   This constraint is essential in the proof that a random graph will not have dense subgraphs. See Proposition 3.1. If it is not satisfied then a random graph is expected to have a graph as dense as the dense subgraph.
5. $\frac{MD^2}{N} \gg (\frac{md^2}{n})^2$.
   When this condition is satisfied, the dense subgraph is hard to detect for both cycle counting (Section 7.2) and SDP (Section 7.3).
6. $d < 2\sqrt{D \log M}$
   This is only needed for Ex-Post detection. If this is not satisfied, $d \geq 2\sqrt{D \log M}$, then for each derivative, the number of defaulted assets exceeds $\frac{D+d}{2}$ with probability at most $1/M$, while the derivatives in the dense subgraph has probability $1/2$ of exceeding the threshold. In an ex post setting, the buyer will see almost all derivatives that exceeds the limit will be derivatives in the dense subgraph, and half of the derivatives in the dense subgraph will exceed the limit. This makes the dense subgraph ex post detectable.
7. $M\sqrt{D} \gg N$
   When this constraint is satisfied, and $b \geq 2\delta\sqrt{\log \frac{MD}{N}}$, the lemon cost for exponential time buyers of binary CDOs will be smaller than $n$, while the lemon cost for polynomial time buyers is always larger than $n$. See Section 5
8. $\log d / \log m < \log D / \log M$ and $\log d / \log n < \log D / \log N$
   When these conditions are satisfied, no algorithm based on *log*-density (such as the algorithm in [6]) can detect the dense subgraph.