Computational Complexity and Information Asymmetry in Financial Products

(Working paper)

Sanjeev Arora[*] Boaz Barak[*] Markus Brunnermeier[†] Rong Ge[*]

February 5, 2012

Abstract

Traditional economics argues that financial derivatives, like CDOs and CDSs, ameliorate the negative costs imposed by *asymmetric information.* This is because securitization via derivatives allows the informed party to find buyers for less information-sensitive 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. Using methods from theoretical computer science this paper shows that derivatives can actually amplify the costs of asymmetric information instead of reducing them. Note that computational complexity is only a small departure from full rationality since even highly sophisticated investors are boundedly rational due to a lack of requisite computational resources.

[*]Department of Computer Science and Center for Computational Intractability, Princeton University, {arora, boaz, rongge}@cs.princeton.edu

[†]Department of Economics and Bendheim Center for Finance, Princeton University, markus@princeton.edu

1 Introduction

Most accounts of the 2007-09 financial crisis (see e.g. [CJS09, Bru09]) agree that mispricing of *financial derivatives* —specifically, structured finance products such as CDOs and CDSs— by market players played a central role. Critics therefore conclude that such derivatives should go through an approval process similar to drugs, which have to be approved by the FDA. Others counter that this would harm the markets.

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 [DeM05] suggests this beneficial effect is large. He shows that even though one would expect that informed sellers would be able to cheat their buyers by suitable "cherry picking" based upon their hidden information, in fact derivatives like CDOs protect the buyers.

The practical downside of derivatives is that they are *complex* assets that are difficult to price. Coval et al. [CJS09] show that pricing (or rating) a structured finance product like a CDO is extremely fragile to modest imprecision in evaluating underlying risks, including systematic risks. Earlier empirical studies suggest that valuations for a given financial product by different sophisticated investment banks can be easily 17% apart [BC97] and that even a single bank's evaluations of different tranches of the same derivative may be mutually inconsistent [Duf07].

Thus structured finance products like CDOs present a theoretical dilemma. Under the usual setup of economic theory, including standard ways of modeling *asymmetric information*, they are provably beneficial, whereas practical difficulties seem to erode some of this benefit.

This paper shows that some of this dilemma can be theoretically captured using the notion of *computational complexity* from computer science. We show that in relatively simple settings akin to those in DeMarzo (2005), real-world buyers will not have enough computational resources for accurate pricing, whereas buyers with unbounded computational resources would be able to price accurately. This can be viewed as an extension of the familiar *bounded rationality idea* [GSe02], with the important difference that this form of bounded rationality applies even to the most sophisticated investment banks since they do not have unbounded computational power.

Our model has several notable features:

1. The largeness of the market—specifically, the fact that sellers are constructing thousands of financial products rather than a single product as was the case in the model of DeMarzo (2005)— allows sellers to cherrypick in such a way that cannot be detected by computationally bounded buyers —i.e., all real-world buyers—whereas it can be detected by computationally unbounded buyers.

   This seems to capture an aspect of financial markets that was often suspected but was difficult to capture in a fully-rational (i.e., computationally unbounded) setting.

   The basic idea is illustrated in a simple setting in Section 2, and exhibited more formally in a DeMarzo-like setting in Section 6.

2. The possibility of cherry picking by sellers creates an Akerloff-like *wedge* between buyer's and seller's valuations of the financial product. We call this the *lemon cost due to computational complexity* and can quantify this wedge for several classes of derivatives popular in securitization. This allows a partial ranking of these classes, which can be seen as a quantification of more familiar heuristic notions of "complexity." This answers the open question of Brunnermeier and Oehmke (2008).

The basic results are stated in Section 2.4, and proved in subsequent sections.

3. The above effects can be obtained in fairly simple models of asset yields, and these models are subcases of more complex models (such as Gaussian copula) used in practice. Note that exhibiting the difficulty of pricing in the subcase is sufficient to exhibit it in more general models. See Section A.

4. It can be difficult for regulatory bodies to control the above-mentioned cherry picking because the cherry picking can be difficult to detect *ex ante.* In some models the cherry picking seems undetectable *ex post.* Both these remain true even in a fully transparenty market where all transactions occur on a public exchange. See Section E.5.

Though the ex-post undetectability is somewhat model-dependent, it does illustrate an interesting possibility. It also implies that verifying the existence of the lemon cost due to computational complexity in historical data (in other words, an empirical test of our paper) may prove difficult. This is even more true since the market has not been fully transparent, and there is no public record of a firm's derivative trades.

Before proceeding with further details we briefly introduce computational complexity and asymmetric information.

Computational complexity and informational asymmetry [1] Computational complexity studies *intractable* problems, those that require more computational resources than can be provided by the fastest computers. A simple example of such a problem is that of *factoring* integers. It's easy to take two random numbers —say 7019 and 5683— and multiply them —in this case, to obtain 39888977. However, given 39888977, it's not that easy to factor it to get the two numbers 7019 and 5683. Algorithm that search over potential factors take a very long time. This difficulty becomes more pronounced as the numbers have more and more digits. Computer scientists believe that factoring an n-digit number requires roughly $exp(n^{1/3})$ time to solve,[2] a quantity that becomes astronomical even for moderate n like 1000. The intractability of this problem leads to a concrete realization of *informational asymmetry.* Anybody who knows how to multiply can randomly generate (using a few coin flips and a pen and paper) a large integer by multiplying two smaller factors. This integer could have say 1000 digits, and hence can fit in a paragraph of text. The person who generated this integer knows its factors, but no computational device in the universe can find a nontrivial factor in any plausible amount of time.[3] This informational asymmetry underlies modern cryptosystems, which allow (for example) two parties to exchange information over an open channel in a way that any eavesdropper can extract *no* information from it —not even distinguish it from a randomly generated sequence of symbols. More generally, in computational complexity we consider a computational task *infeasible* if the resources needed to solve it grow *exponentially* in the length of the input, and consider it *feasible* if these resources only grow polynomially in the input length.

[1]Computational complexity has been applied to economics before. Postlewaite et al refer to the possibility of learning via computation as fact-free learning. The field of algorithmic game theory studies how game theoretic results are affected when agents have bounded computational power.

[2]The precise function is more complicated, but in particular the security of most electronic commerce depends on the infeasibility of factoring integers with roughly 800 digits.

[3]Experts in computational complexity should note that we use factoring merely as an simple illustrative example. For this reason we ignore the issue of *quantum computers*, whose possible existence is relevant to the factoring problem, but does not seem to have any bearing on the computational problems used in this paper.

Computational complexity immediately implies the existence of hard-to-price derivatives. 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 Goldman Sachs would have no idea what to pay for it. This example shows both the difficulty of pricing arbitrary derivatives and the possible increase in asymmetry of information via derivatives.

While this "factoring derivative" is obviously far removed from anything used in current markets, in this work we show that similar effects can be obtained in simpler and more popular classes of derivatives that are essentially the ones used in real life in securitization of mortgages and other forms of debt.

We will measure the cost of complexity by resorting to an Akerloff-like notion of *wedge* between the value of the asset to the buyer and the seller. The *lemon cost of complexity* is estimated 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.

2 The basic model

DeMarzo (2005) (and earlier, DeMarzo and Duffie (1999)) considers a simple model of asymmetric information and shows how CDOs can help lower the "Akerloff wedge" due to asymmetric information. We adopt a similar model of asymmetric information, but the market has a couple of important differences.

DeMarzo assumes that a seller has N assets, and the yield Y_i of the ith asset is $X_i + Z_i$ where both X_i, Z_i are random variables whose distributions are known to seller as well as buyer. At the start, seller has seen the value of each X_i and buyer hasn't —this is the asymmetric information. Seller prefers cash to assets, since his discount rate is higher than that of the buyers. If the seller were to sell the assets directly, he can only charge a low price since potential buyers are wary that the seller will offer primarily lemons (assets with low X_i) to the buyer. DeMarzo (2005) shows that it is optimal for the seller to first bundle the assets and then tranche them in a *single* CDO. The seller offers the buyer the senior tranche and the retains the riskier junior tranch. Since the seller determines the threshold of the senior tranche, he determines the faction of the cash flow stream he can sell off. Selling off a smaller fraction is costly for the seller, but it signals to the buyer that his private information $\sum_i X_i$ is high. Overall, tranching leads to a price at which seller is able to sell his assets at a better price because the difference between buyer's and seller's valuations (the "lemon wedge", which has to be overcome by buyer's preference for cash) goes to 0 on average as $N \to \infty$.

Our model can be phrased in DeMarzo's language, but differs in two salient ways. First, we assume that instead of N assets, there are N asset classes where seller holds C iid assets in each class. (It is customary in securitization to think of assets as being bunched into classes of similar assets, eg, loans made in NJ to borrowers whose credit score lies in a certain range.) Some classes are "lemons", and these are drawn from a distribution known both to seller and buyer. These have significantly lower expected yield. Seller knows the identity of lemons, but buyers only knows the prior distribution.

The second major difference in our model is that we assume that instead of selling a single CDO, the seller is offering M CDOs. Now buyer (or buyers) must search for a "signal" about seller's private valuation by examining the M offered CDOs. DeMarzo's analysis has no obvious extension to this case because this signal is far more complex than in the case of a single CDO, where all assets have to be bundled into a single pool and the only parameter under seller's control is the threshold defining the senior tranche.

We will show that if buyers are fully rational and capable of exponential time computations, then DeMarzo's essential insight (and conventional wisdom) about CDOs can be verified: lemon costs do get ameliorated by CDOs. However, if buyers are computationally bounded, then under reasonable assumptions about their computational power we can show that lemon costs do not get ameliorated by CDOs for interesting model parameter choices.

2.1 An illustrative example

Now we illustrate our main points with a simple example. As already mentioned, our more general results basically embed this example in a more complicated setting.

Consider a seller with N asset classes, each of which contains C assets. Some asset classes are "lemons": assets in these classes will always *default* and have payoff 0. All other asset classes are good: assets in these classes pay $1/C$ with probability $1/2$ and default (i.e., have payoff 0) with probability $1/2$. The assets between different asset classes are independent.

The buyer's prior is that the number of lemon classes is uniformly distributed in $[0, 2n]$, and the set of lemon classes is uniformly picked from among all classes. However, the seller has additional information: he knows the number of lemon classes as well which classes are lemons. This is the asymmetric information.

Since the expected number of lemon classes is n, each with payoff 0, and the remaining $N - n$ good classes have payoff $1/2$, a risk-neutral buyer purchasing the entire portfolio would be willing to pay the expected yield, which is $(N - n)/2$. Thus a *wedge* a la Akerloff occurs for sellers who discover that the number of lemons is lower than the expectation. They might exit the market, or buy anyway if they prefer cash by an amount that overcomes the wedge.

Conventional wisdom (first exhibited in a formal model by DeMarzo) holds that structured finance potentially allows the seller to turn (most of his) assets into cash while incurring a smaller wedge. For instance, he could securitize the assets into a single Collateralized Debt Obligation (CDO), and sell the senior tranche. The yield of the senior tranche is less dependent on the number of lemons, and thus less information-sensitive.

In this paper we are primarily interested in the case where the number of assets held by the seller is *large*, and so they cannot be packaged into a single CDO. Instead he must partition them into multiple CDOs. Clearly, he can use his private information in constructing these CDOs.

In principle derivatives still allow the lemon wedge due to asymmetric information to be ameliorated. Consider the following: seller creates M new financial products, each of them depending on a pool of D of the underlying assets. We assume $MD = NC$ and that every asset occurs in exactly one pool. If the assets are randomly partitioned into the pools, then the lemon assets are quite likely to be well-spread among the pools: in the expectation only half of the D assets in the pool will default, and the standard deviation is $\sqrt{D}/2$, which is much smaller. Thus the financial product based upon that pool has a payoff of $N/3M$ if the number of lemons exceeds $D/2$ by at most t standard deviations, and otherwise has payoff 0. If t is moderately large, then by the strong law of large numbers, Pr[a pool has at most $D/2 + t\sqrt{D}$ lemons] is quite close to 1, and thus the value of the entire portfolio, denoted V is close to $N/3M \times M = N/3$.

Henceforth we call such a product a "Binary CDO"; it can be viewed as the senior tranche of a simple CDO[4].

If the seller indeed picks the pools randomly —i.e., the entire portfolio of assets is randomly partitioned into the M pools— then one can check (see Section 3) that the portfolio's expected yield is only mildly affected by the presence of lemons. If V is the expected yield when the number of lemon classes is 0, then the yield is still $V - o(n)$ when the number of lemon classes is $2n$, the maximum possible. (Here $o(n)$ means some function of n that grows slower than n; the exact function appears in Section 3.) In other words, the lemon wedge is $o(n)$ instead of n, and thus derivatives have helped significantly reduce the lemon wedge. Thus the seller is in principle able to sell off the least information-sensitive portion of the risk.

However, the above description assumed that seller creates the pools *disinterestedly* using pure randomness. But this may be against his self-interest given his secret information! Assuming the buyer pays the same price as long as the seller does not tamper too much with the pool structure, some calculations show that his optimum strategy is to pick some subset of m of the financial products, and ensure that the lemon assets are overrepresented in their pools—to an extent about $\sqrt{D}$, the standard deviation, just enough to significantly skew the probability of default. We call this subset of CDOs the "boobytrap." Thus the CDOs in the boobytrap have a significantly higher probablity of default than buyers expect, whereas the remaining CDOs have a slightly lower number of lemons (and hence default probability) than buyers expect. Nevertheless, a simple calculation shows (see Section 3) that the net effect is that the expected yield of the entire portfolio is lower by a noticeable extent, which benefits the seller if the buyer does not notice this booby trap and still pays $V - o(n)$.

Can sellers profit from such cherrypicking (possibly blaming random chance when buyers experience losses at the end)? We now show that fully informed and computationally unbounded buyers will not be fooled. We will assume that all transactions –and hence the composition of CDOs— is public and visible to all buyers, so seller cannot fool an individual buyer by showing each only a part of the portfolio[5].

Fully rational (i.e., computationally unbounded) buyer: He will not be fooled. Even though he doesn't know the set of lemon classes, he knows that even in a randomly chosen portfolio the possibility of setting up such a boobytrap is vanishingly remote. Therefore it suffices for him to rule out the existence of any boobytrap as follows: he enumerates over *all* possible $2n$-sized subsets of $[N]$ and verifies that *none* of them are over-represented in *any* subset of m (or even m/10) products. If so, he can conclude that no booby trap exists in the presented portfolio of CDOs even though he does not know the identity of the lemon classes.

Thus fully rational buyers will only be willing to buy from sellers whose portfolio of binary CDO does *not* have a boobytrap, in other words, have value at least $V - o(n)$. Thus the lemon wedge is greatly ameliorated if buyers are fully rational[6]

Real-life buyer who is computationally bounded: For him the above computation for detecting booby traps is infeasible even for moderate parameter values. In fact, the problem of detecting a boobytrap is equivalent to the so-called *hidden dense subgraph* problem, which computer scien-

[4]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 Appendix A.

[5]This transparency does not currently hold in the derivatives market, but may be a reasonable approximation if buyers are well-informed. In any case, the point of our paper is to exhibit difficulties of derivative pricing even under idealized conditions.

[6]Note that this result also precludes sellers who cherrypick and later claim that the bad returns happened "by chance", since the probability of a boobytrap appearing this way is extremely close to 0.

tists believe to be intractable (see discussion below in Section 2.4). Moreover, under reasonable computer science assumptions, there is a way for the seller to "plant" a boobytrap in a way that the resulting pooling will be *computationally indistinguishable* from a random pooling. Even quite large boobytraps may be undetectable: the expected yield of the entire portfolio could be much less than say $V - 4n$ and yet the buyer may not be able to distinguish it from a truly random (i.e., honestly constructed) portfolio, whose yield is $V - o(n)$. If buyers are computationally bounded, then introducing derivatives into the picture not only fails to reduce the lemon wedge, but paradoxically, *amplifies* it beyonds its *a fortiori* maximum of $2n$!

Ex ante and Ex post Detection In most parts of the paper we discuss *ex ante* detection, where the buyers try to detect the hidden dense subgraph before the outcomes have been revealed. We shall show that for computationally limited buyers it is hard to distinguish between a booby-trapped derivative and a truly randomly generated derivative (see Section E). To avoid the hardness of ex ante detection, the buyers can try to find the dense subgraph after the outcomes of assets are revealed (which we call *ex post* detection). Ex post detection is clearly easier than ex ante detection as the buyers realize that the financial products had a much lower yield than expected and have more information. However in certain models even ex post the buyers would be unable to prove that the seller has tampered the derivatives by finding a dense subgraph, see Section E.5 in the appendix.

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 portfolio of financial products is *untampered* i.e., free of booby traps. (Computer scientists believe that there is no simple way to prove the absence of a dense subgraph; this is related to the $NP \neq coNP$ conjecture). Nevertheless, we do show in Section 7 that one could use Computer Science ideas in designing derivatives that are tamperproof in our simple setting.

Relationship to the known "sensitivity" problems of CDOs. It is known that CDOs are extremely sensitive to small changes in the distribution of yields of the underlying assets. Coval et al. [CJS09] suggest that such "modeling problems" may underlie the mispricing of CDOs that led to the 2008 crash. The key difference here is that the buyers *do* know the distribution of the yields; modeling error cannot be blamed. Nevertheless the seller is able to exploit the sensitivity properties of CDOs while keeping such manipulation hidden from computationally limited buyers.

2.2 Lemon Cost

Now we formally define *lemon cost*, a way to quantify the sensitivity of the expected value of a financial product to the presence of *lemon* assets. The identity of these lemons is known to the seller but not to the buyer. As in the illustrative example, we assume that the seller has to use all his assets in the portfolio of products he sells, so the only way for seller to benefit from his secret information (i.e., to cherrypick) is to *place* the lemon assets in the financial product in a way that maximizes his expected return.

As in the illustrative example, we assume that assets are drawn from N asset classes, where each class has C assets with iid yields. Some classes are lemons, and we assume for simplicity that the number of lemon classes is uniformly distributed in $[0, 2n]$. Intuitively, the lemon cost of a financial product (or portfolio of products) referencing these assets is the difference between the yield when the number of lemons is 0 and the yield when this number is $2n$, assuming optimum cherry-picking by seller. (Recall that the seller's profit equals the difference between the payoff of the underlying assets and the payoff of the financial products. Since the expected payoff of assets is a fixed value known to the seller, maximizing the expected profit is equivalent to minimizing the expected payoff of the products he sells to the buyers.)

The input to the financial product is a matrix $A \in \{0,1\}^{N\times C}$, where $A_{i,j}$ indicates whether the j-th asset in i-th asset class defaulted. The financial product is a function F mapping A to a yield $F(A)$. The input A should be chosen from certain distributions. Let $\mathcal{D}_l$ to be a set of distributions of A, where the number of lemons is l. If distribution $D \in \mathcal{D}_l$, then D samples each row of A independently, exactly l rows will be the all 1's vector (these are the lemon classes), and for the other $N - l$ rows, each $A_{i,j}$ is chosen uniformly random from $\{0,1\}$. Thus any distribution $D \in \mathcal{D}_l$ corresponds to a situation that some fixed l asset classes are lemons (all assets in those classes will default), and all other asset classes are good classes (each asset defaults with probability 1/2).

When there are no lemons, the matrix A is uniformly sampled from $\{0,1\}^{N\times C}$. In this case there's only one distribution in the class $\mathcal{D}_0$, and we call it D_0. The expected yield of the financial product is $\mathrm{E}_{A\sim D_0}[F(A)]$.

When there are $2n$ lemon classes, the seller who knows their identity will try to rearrange the asset classes, so that the expected yield of the derivative is minimized. That is, the expected yield will be

$$\min_{D\in\mathcal{D}_{2n}} \mathop{\mathrm{E}}_{A\sim D}[F(A)].$$

Here taking the minimum assumes that seller optimally places the lemon assets (indeed, each distribution in D_l corresponds to a way to place the lemon assets). According to Akerlof [Ake70], the sellers with the best quality assets (no lemons) will exit the market unless they wish to bear a wedge that is equal to the difference in the expected yields, and we define this wedge as the lemon cost $\Delta_{F,n}$.

$$\Delta_{F,n} = \mathop{\mathrm{E}}_{A\sim D_0}[F(A)] - \min_{D\in\mathcal{D}_{2n}} \mathop{\mathrm{E}}_{A\sim D}[F(A)]. \tag{1}$$

2.3 The Cost of Complexity

The *cost of complexity* arises from the possibility that computationally limited buyers will be unable to distinguish between two different financial products even though they have very different expected yield and lemon cost. When offered one of these two products, it would be logical for a buyer to assume that it is the one whose yield is lower (which usually means higher lemon cost). Thus the buyer's computational limitation introduces another Akerloff-like wedge, which is captured by the cost of complexity.

Formally, it does not make sense to talk about the computational complexity of computing the yield of a single financial product. Instead one talks about the complexity of computing one function drawn from an ensemble (or distribution) of financial products. The above illustrative example implicitly refered to two distributions of financial products: the randomly-constructed CDO portfolio in which assets are randomly distributed among the M CDOs, and the *boobytrapped* variation of this portfolio, where the boobytrap consists of a randomly selected subset of CDOs. As is customary in study of computational complexity, we rely on asymptotic analysis and assume M is large.

In the rest of the paper, a *computationally limited buyer* is synonymous with the computer science notion of "polynomial-time computer" as explained in the Introduction. For two distributions of financial products $\mathcal{R}$ and $\mathcal{P}$, we say they are ϵ-computationally indistinguishable ($\mathcal{R} \approx_\epsilon \mathcal{P}$) if the following is true: whenever a computationally limited buyer tries to compute the expected yield of a product drawn from $\mathcal{R}$ or $\mathcal{P}$, his answer differs by less than ϵ in the expectation. Consequently, his calculation of the lemon cost Δ will also differ by less than 2ϵ.

Throughout the paper we will always assume $\mathcal{R}$ is the "canonical" random construction of the financial products. This reflects the traditional belief that financial products such as CDOs are constructed by randomly throwing together diverse assets, so that law of large numbers applies. In our illustrative example above, $\mathcal{R}$ corresponds to partitioning the portfolio of assets randomly (i.e., disinterestedly) into pools. The distribution $\mathcal{P}$ corresponds to a CDO portfolio in which the seller has planted a "booby trap" in a random subset of CDOs.

For any class $\mathcal{F}$ of derivatives (e.g. CDO, CDO-squared, etc.), the *cost of complexity* CoC_ϵ is the maximum difference between the true lemon cost of two distributions $\mathcal{R}$ and $\mathcal{P}$ over $\mathcal{F}$ that are ϵ-indistinguishable:

$$CoC_\epsilon(\mathcal{F}, n) = \max_{\mathcal{P} \approx_\epsilon \mathcal{R}} \mathop{\mathrm{E}}_{F \sim \mathcal{P}}[\Delta_{F,n}] - \mathop{\mathrm{E}}_{F \sim \mathcal{R}}[\Delta_{F,n}].$$

Examining the definition of lemon costs in (1) we see that the distribution D_0 is common to the two terms and gets canceled out. Thus $CoC_\epsilon(\mathcal{F}, n)$ is nothing but the difference between expected yields of products drawn from $\mathcal{R}$ and $\mathcal{F}$ when the number of lemons is $2n$. We chose to define cost of complexity using lemon costs because lemon cost is the more intuitive measure of effects of asymetric information, and we are trying to quantify the effect of computational complexity on this measure.

2.4 Our Results

Theorem 1. *Assume that the parameters N, M, D, n, m, d are such that the distributions $\mathcal{R}$ and $\mathcal{P}$ are computationally indistinguishable, the sellers can construct derivatives so that the lemon costs of different types of the derivatives are as listed in Tabel 1.*

Our results are summarized in Table 1, which lists the lemon cost for different types of financial products. To highlight the effect of the assumption about buyers having bounded computational power ("feasibly rational"), we list two values for each type of financial product. The readers should keep in mind that fully rational buyers can distinguish whether the derivatives have a booby trap or not, and therefore make sure to have a lemon cost as in the "no booby trap" case, but "feasibly rational" buyers cannot distinguish between the "no booby trap" and "booby trap" case, and cannot avoid to suffer a lemon cost as in the "booby trap" case. Unsurprisingly, without derivatives the buyer incurs a lemon cost of n. In the "Binary CDO" setting described in the illustrative example, things become interesting. It turns out that using exponential time computation the buyer can verify that the CDO was *properly* constructed, in which case the cost to the buyer will be actually much smaller than n, consistent with the received wisdom that derivatives can insulate against asymmetric information. But, under the computational complexity assumptions consistent with the current state of art, if the buyer is restricted to feasible (i.e., polynomial time) computation, then in fact he *cannot verify* that the CDO was properly constructed. As a consequence the cost of the n junk assets in this case can in fact be much larger than n. In a CDO^2 (a CDO comprised of CDO's, see Section 5) this gap can be much larger with essentially zero lemon cost in the exponential case and maximal cost in the polynomial case.

Model	**Lemon cost**	**Reference**
Derivative-free	n	
binary CDO, no booby trap	$\sim n(N/M\sqrt{D}) \ll n$	Theorem 3
binary CDO, booby trap	$\sim n\sqrt{N/M} \gg n$	Theorem 3
tranched CDO, no booby trap	$\sim n(N/MD)$	Theorem 4
tranched CDO, booby trap	$\sim n(\sqrt{N/MD})$	Theorem 4
binary CDO^2, no booby trap	0	Theorem 5
binary CDO^2, booby trap	$N/4$	Theorem 5
tranched CDO^2, no booby trap	0	Theorem 6
tranched CDO^2, booby trap	$\sim n(\sqrt{N/MD})$	Theorem 6

Table 1: Cost of n junk assets (asset classes) in different scenarios, for N assets, and CDO of M pools of size D each. $\sim$ denotes equivalence up to low order terms. 0 means the term tends to 0 when N goes to infinity. See the corresponding theorems for the exact setting of parameters. Fully rational buyers will be able to distinguish "booby trap" and "no booby trap", but computationally limited buyers cannot distinguish the two cases.

2.5 Parameters and Notations

In this paper we use *asymptotic analysis* as in most computer science research. Here we define some of the notions:

Definition 1. For functions $f(n)$ and $g(n)$, we say $f(n) = O(g(n))$ if there exists some constant $C > 0$ such that $f(n) \leq Cg(n)$. Similarly, $f(n) = \Omega(g(n))$ if there exists some constant $C > 0$ such that $f(n) \geq Cg(n)$. The notation $f(n) = \Theta(g(n))$ if both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, that is, there are constants $0 < C_1 < C_2$ such that $C_1 g(n) \leq f(n) \leq C_2 g(n)$.

The big-O, Θ, Ω notations are commonly used in computer science literatures. When $f(n) = \Theta(g(n))$, the functions $f(n)$ and $g(n)$ are roughly equivalent. Although theoretically the constants in the definitions are arbitrary and might be unrealistically large or small, in this paper all the constants are within reasonable range and most asymptotic effects are already significant at magnitude of thousands.

We also use the small-o and ω notations:

Definition 2. For functions $f(n)$ and $g(n)$, we say $f(n) = o(g(n))$ if for any constant $C > 0$, there exists a constant $n(C)$ such that when $n > n(C)$, $f(n) < Cg(n)$. Similarly, $f(n) = \omega(g(n))$ if for any constant $C > 0$, there exists a constant $n(C)$ such that when $n > n(C)$, $f(n) > Cg(n)$.

If $f(n) = o(g(n))$, we think of $f(n)$ as negligible when comparing to $g(n)$.

Finally we introduce some non-standard notations, including $\tilde{O}$ (ignore log factors) and $\gg$ (much greater than):

Definition 3. $f(n) = \tilde{O}(g(n))$ iff $f(n) = O(g(n) \log g(n))$. $f(n) = \tilde{\Omega}(g(n))$ iff $f(n) = \Omega(g(n)/\log g(n))$. $f(n) = \tilde{\Theta}(g(n))$ iff $f(n) = \tilde{O}(g(n))$ and $f(n) = \tilde{\Omega}(g(n))$.

In this paper, when we say $f(n) \gg g(n)$ we mean $f(n) = \Omega(g(n)N^{\epsilon})$ for some positive constant ϵ. Here N always denote the number of asset classes throughout the paper.

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

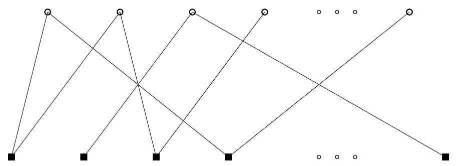

Figure 1: Using a bipartite Graph to represent asset classes and derivatives. There are M vertices on top corresponding to the derivatives and N vertices at the bottom corresponding to asset classes. Each derivative references D assets in different classes.

("bottom") side, and top degree D. It is convenient to view the relationship of derivatives and asset classes as a *bipartite graph*, see Figure 1. Derivatives and asset classes are vertices, with an edge between a derivative and an asset classes if the derivative depends on an asset in the class. 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.

Densest subgraph problem. 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 $\mathcal{P}$ 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 $|S| = 2n, |T| = m$, and then choosing D random neighbors for every vertex outside of T[7], 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 [ABW10] as a source for public key cryptography[8]. The state of art algorithms for both the worst-case and average-case problems are from a recent paper of Bhaskara et al [BCC$^+$09]. 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 δ, $N = o(MD)$, $(md^2/n)^2 = o(MD^2/N)$, $n = \Omega(N^{0.5+\delta})$, $m = \Omega(M^{0.5+\delta})$, $d = \tilde{O}(D^{0.5})$ then there is no $\epsilon > 0$ and poly-time algorithm that distinguishes between $\mathcal{R}$ and $\mathcal{P}$ with advantage ϵ.

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 $m = \tilde{O}(n\sqrt{M/N})$.

[7]Actually, for vertices outside of T, we on average choose $(nMD - Nmd)/N(M - m)$ neighbors in S to make the degree distributions similar for vertices in S and outside S. But since normally $nMD \gg Nmd$, in some calculations we safely assume that all the D neighbors are uniformly random

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

3 Lemon Cost Bounds for Binary Derivatives

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

Recall the model we used in the illustrative example: there are N *asset classes* (e.g., the subprime mortgages in California) and each class has $C = MD/N$ assets. Each asset class has a parameter p, which indicates the general performance of the asset class. Conditioned on the performance value p, each asset in this asset class will default (give 0 return) with probability $1-p$, and give return $1/C$ with probability p (therefore the asset class has expected return p). In the illustrative example, the p values for good asset classes are $1/2$, and the values for lemon classes are 0. In this section we continue to use this simple assumption, but notice that the same asymptotic result holds for a much broader choice of parameters. In fact, as long as the expected p value for good classes is significantly larger (i.e. larger by at least a constant) than the expected p-value for lemons, the theorems in this section will still hold.

In our setting the seller generates M binary derivatives, where the value of each derivative is based on the D of the assets from D different asset classes. There is some agreed threshold value b, 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 $V = \frac{D-b}{2D}\frac{N}{M}$ (this amount is equivalent to the yield of the $\frac{D-b}{2}$ assets that have not defaulted so the seller can always use the yields of assets to pay for the derivatives).

Since each derivative depends on D independent assets, each of which has probability 1/2 of defaulting, 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 Φ 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 $M\frac{D-b}{2D}\frac{N}{M} \sim N/2$. Note that this calculation is the same regardless of which assets are chosen for the derivative as long as they come from different asset classes.

We now compute the effect of $2n$ lemon classes on the value of all the derivatives. In this case the shape of the pooling *will* make a difference. As we explained, the shape of the pooling is represented using a bipartite graph. The top M vertices correspond to the derivatives, the bottom N vertices correspond to the asset classes, each top vertex will have D edges connecting to the asset classes. Remember that this is what we called an (M, N, D) graph.

To increase his expected profit seller can carefully design this graph, using his secret information. The key observation is that though each derivative depends upon D assets, in order to substantially shift 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 ith derivative. Since each of the $2n$ junk asset classes contains $C = MD/N$ junk assets, we have $\sum_{i=1}^{M} t_i = \frac{2nMD}{N}$. We call this derivative

F, then in this case the lemon cost $(E_{A\sim D_0}F(A) - E_{A\sim D_{2n}}F(A))$ 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 Section B.1.)

Therefore the lemon cost is maximized by choosing some m derivatives and letting each of them have at least $d = k\sqrt{D}$ assets from the set of junk asset classes. In the bipartite graph representation, this corresponds to a dense subgraph, which is a set of derivatives (the manipulated derivatives) and a set of asset classes (the junk classes) that have more edges between them than expected. This precisely corresponds to the pooling graph containing an $(m, 2n, d)$ 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.

Note that even random graphs have dense subgraphs. For example, when $md = n$, any (M, N, D) 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 the following theorem, proven in Section B.2, shows:

Theorem 2. *When $n \ll md$, $\frac{dN}{Dn} > (N + M)^{\epsilon}$ for some constant ϵ, 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 $(m, 2n, d)$-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 $\Phi'(-b/2\sigma)\frac{2nD}{2N}\frac{1}{\sigma} \cdot N/2 = O(e^{-(b/2\sigma)^2/2}n\sqrt{D})$ which can be made $o(n)$ by setting b to be $\Theta(\sqrt{D \log D})$. The effect of an $(m, 2n, d)$ subgraph on the lemon cost is captured by the following theorem (whose proof is deferred to Appendix B.2):

Theorem 3. *When $b > 3\sqrt{D}$, $d - b > 3\sqrt{D}$, $n/N \ll d/D$, a planted (m, n, d) subgraph will generate an extra lemon cost of at least $\Omega(mV)$. The Cost of Complexity for portfolios of binary derivatives is at least $\Omega(n\sqrt{N/M})$ under the densest subgraph assumption.*

Assume $M \ll N \ll M\sqrt{D}$ and set $m = \tilde{\Theta}(n\sqrt{M/N})$ so that a graph with a planted $(m, 2n, d)$ subgraph remains indistinguishable from a random graph under the densest subgraph assumption. Setting $b = 2\sigma\sqrt{\log \frac{MD}{N}}$ the theorem implies that a fully rational buyer can verify that nonexistence of dense subgraphs and make sure the portfolio has a lemon cost of $n\frac{N}{2M\sqrt{D}} = o(n)$, while a polynomial time buyer will have to bear with a lemon cost of $n\sqrt{N/M} = \omega(n)$.

4 Non-Binary Derivatives and Tranching

We showed in Section 3 that the lemon cost of binary derivatives can be large when computational complexity is considered. However, binary derivatives are less common than securities that use *tranching*, such as CDOs (Collateralized debt obligations). In a normal securitization setting, the seller of assets (usually a bank) will offload the assets to a shadow company called the *special purpose vehicle* or SPV. The SPV recombines these assets into notes that are divided into several tranches ordered from the most senior to the most junior (often known as the "equity" or "toxic waste" tranche). If some assets default, the junior-most tranche takes the first loss and absorbs all losses until it's "wiped out" in which case losses start propagating up to the more senior tranches. Thus the senior-most tranche is considered very safe, and may even receive a AAA rating.

For simplicity here we consider the case where there are only two tranches, senior and junior. If the percentile of the loss is below a certain threshold, then the junior tranche suffers the entire loss; if the loss is above the threshold, then the junior tranche lose all its value and senior tranche is responsible for the rest of the loss. Clearly, the senior tranche has lower risk than the whole asset. We will assume that seller the retains the junior tranche, which is normally believed to signal his belief in the quality of the asset. Intuitively, such tranched CDOs should be less vulnerable to manipulation by seller compared to binary CDOs, and we'll quantify that this is indeed true. Nevertheless, we show that our basic result in Section 3 —that the lemon cost is much larger when we take computational complexity into account—is unchanged.

We adapt the parameters and settings in Section 3, the only difference is that we replace our binary derivative with the senior tranche derivative. In the event of $T > \frac{D+b}{2}$ defaults, instead of payoff of 0 that the binary CDO had, this one has a payoff of $D - T$ assets. (In the other case it has a payoff of $V = \frac{D-b}{2}$ assets as in the binary case). Without lemons, the expected value of this derivative is approximately the same as the binary one, as the probability of $T > \frac{D+b}{2}$ is very small.

The first observation regarding tranching is that in this case the lemon cost can never be larger than n, since the default of a single asset can cause a combined loss of at most one unit to all the tranches. In fact we will show that in this case for derivatives constructed with either the random graph ($\mathcal{R}$) or the graph with planted dense subgraph ($\mathcal{P}$), that the lemon costs are *less* than n. But there will still be a difference between the two cases — a cost of δn in the exponential-time case and $\sqrt{\delta}n$ in the polynomial-time case, where $\delta \sim \frac{N}{MD}$. This is shown by the following theorem (whose proof is deferred to Appendix C):

Theorem 4. *When $b \geq 3\sqrt{D}$, $d-b \geq 3\sqrt{D}$, $d < \sqrt{D}\log D$, $n/N \ll d/D$, the lemon cost for a graph in distribution $\mathcal{P}$ (a graph with $(m, 2n, d)$ subgraph) is $\epsilon n + \tilde{\Theta}(\frac{\sigma}{D}mV)$, where $\epsilon = O(be^{-(b/2\sigma)^2/2}/\sigma)$. The Cost of Complexity of CDOs is at least $\Omega(n\sqrt{N/MD})$ under the densest subgraph assumption.*

Therefore in CDOs, setting b sufficiently large the lemon cost for graphs with planted dense subgraph is $\Theta(n\sqrt{N/MD})$, and computationally limited buyers will have to bear it because they cannot distinguish it from a random generated graph. The lemon cost for exponential time buyers is $\Theta(n \cdot N/MD)$. Since $MD > N$, the lemon cost for polynomial time buyer is always larger. Nevertheless, the gap in lemon cost for tranched CDO's is smaller than the gap for the binary CDO.

5 Lemon Cost for CDO^2

In this section we consider the cost of complexity for a more complex derivative called CDO^2. Simply speaking, a CDO^2 is a CDO of CDOs: it treats the tranches of CDOs as assets (sometimes called "inner" CDOs), and on top of them builds a new CDO structure (which is called "master" or "outer" CDO). By similar methods we can treat CDO^3, which are CDOs of CDO^2, and more generally CDO^n.

In Sections 3 and 4, we have analyzed bundles of CDOs. In this section we shall collect all the CDOs from these bundles (the inner CDOs) and construct a single CDO^2 portfolio (the outer CDO). When the thresholds are chosen correctly we shall see that these CDO^2 portfolios are much more vulnerable to the seller's tampering than the original bundles of CDOs, and the effect is more pronounced if the CDO^2 involves binary CDO's (which were described in Section 3). We defer the proofs of the theorems below to Appendix D .

We will continue to assume the intractability of the Densest subgraph problem with the basic parameters N, n, M, D, m. To simplify exposition we set $M = N^{0.8}$, $D = N^{0.3}$, $n = N^{0.8}$, $m = N^{0.7}$, $d = 6\sqrt{D}$; this set of parameters seems difficult for current algorithms.

First we consider binary CDO^{2}s, which are easier to analyze. The binary CDO^2 takes a set of binary CDOs as described in Section 3, and combines them to make a single derivative. The first level of binary CDOs (inner CDOs) have a payoff threshold set at $\frac{D+b}{2}$ defaults where $b = 3\sqrt{D}$; above this the payoff is 0. We say an inner CDO defaults if it gives 0 payoff. The outer part of CDO^2 combines M of these inner CDOs. The whole CDO^2 portfolio pays $V = N/4$ when the number of defaulted inner CDOs is less than a threshold T. When at least T inner CDOs default, the CDO^2 pays nothing. This threshold T will be set so that the CDO^2 is securitized and can be paid for out of the seller's profits from the N assets. Notice that the inner CDOs and the outer CDO are all "binary": they will generate a fixed amount of payoff whenever the number of defaults is below a certain threshold and pay nothing if the threshold is exceeded.

From Section 3 we know when the bundle of inner CDOs have no "booby traps" (fully random case, distribution $\mathcal{R}$), the expected number of defaulted inner CDOs is small. On the contrary, when the seller plants a "booby trap" in the bundle of inner CDOs (booby-trapped case, distribution $\mathcal{P}$) the expected number of defaulted inner CDOs is much larger. We denote the former as E[random] and the latter as E[planted]. The following theorem shows when the threshold of the outer CDO is set appropriately between these two numbers, the cost of complexity for binary CDO^2's can be really high.

Theorem 5. *In the binary CDO^2 setting, when $m^2 \gg \frac{M^2D^2}{N}$, $T = \frac{1}{2}(\mathrm{E}[random] + \mathrm{E}[planted])$, the lemon cost for derivatives in $\mathcal{P}$ is $(1 - o(1))N/4$, while the lemon cost for derivatives in $\mathcal{R}$ is only $N/4e^{-m^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant. That is, the Cost of Complexity for binary CDO^{2}s is at least $\Omega(N)$.*

Again, commonly used CDO^{2}s are usually not binary and involves tranche structrues as in Section 4. We shall take the CDO bundle we constructed in Section 4 and make a CDO^2 out of the senior tranches.

Recall that the inner CDOs are divided into two tranches as in Section 4, the senior tranches are less risky because they will always pay $V = \frac{D-b}{2}$ when there are less than $\frac{D+b}{2}$ defaults and their values only decrease linearly even if more than $\frac{D+b}{2}$ defaults occur. To add another layer of protection, the sellers collect all the M senior tranches of the CDOs to construct a CDO^2 portfolio. The maximum payoff of this portfolio is $V_{CDO^2} = N\frac{D-b}{2D}$ since there are M senior tranches and each tranche has maximum payoff $\frac{D-b}{2D}\frac{N}{M}$.

The seller shall choose a threshold T for the outer CDO to separate it into two tranches. The junior tranche is responsible for the first T of the loss and the senior tranche is responsible for the rest $V_{CDO^2} - T$.

Suppose the senior tranches of inner CDOs pays S. When this payoff value S is more than $V_{CDO^2} - T$, the senior tranche of the outer CDO pays its full value $V_{CDO^2} - T$ and all the loss are in the junior tranche of the outer CDO; when the payoff value $S < V_{CDO^2} - T$, the senior tranche of the outer CDO begins to suffer loss, and its payoff is equal to S. The seller shall claim that the senior tranche of the outer CDO is really riskless because it only starts suffering loss when many senior tranches of inner CDOs default, which are themselves low probability events. The buyers will buy the senior tranche of the outer CDO and the seller retains all the junior tranches of the inner CDOs and the outer CDO.

As we proved in Section 4, when the bundle of inner CDOs is constructed according to distribution $\mathcal{R}$ (fully random), the expected loss for the senior tranches of inner CDOs is very small. On the other hand, when the bundle of inner CDOs is constructed according to distribution $\mathcal{P}$ (booby-trapped), the expected loss for the senior tranches of inner CDOs is significantly larger. Use E[random] to denote the expected loss in the senior tranches of inner CDOs when the graph is chosen from $\mathcal{R}$, E[planted] to denote the expected loss in the senior tranches of inner CDOs when the graph is chosen from $\mathcal{P}$, we have the following theorem.

Theorem 6. *In the CDO^2 with tranching case, when $m^2d^2 \gg M^2D^2/N$, $T = \frac{1}{2}(\mathrm{E}[random] + \mathrm{E}[planted])$, the lemon cost for derivatives in distribution $\mathcal{P}$ is $\Theta(mdN/(MD))$, while the lemon cost for derivatives in distribution $\mathcal{R}$ is at most $mdN/MDe^{-m^2d^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant. That is, the Cost of Complexity for CDO^2's is at least $\Omega(n\sqrt{M/N})$ under the densest subgraph assumption.*

In conclusion, for both binary CDO^{2}s and CDO^{2}s with tranching, the lemon cost for graphs with planted dense subgraph can be exponential times larger than the lemon cost for random graphs. In particular, for polynomial time buyers who cannot detect the booby trap, the lemon cost and cost of complexity of binary CDO^{2}s can be as large as $N/4$— a large fraction of the total value of the asset.

6 Buyer's Reaction and Liquidity Costs

DeMarzo [DeM05] explains the role of CDOs in ameliorating lemon costs or *liquidity costs* incurred by a seller who tries to sell a security about which he/she has private information. His model takes into account strategic behavior by buyer and seller (e.g., given that seller offers to sell the buyer a fraction of a CDO, how should buyer interpret this as a "signal" of the seller's private information?). Now we show that the lemon cost due to computational complexity also exists in a formal model similar to DeMarzo's. If the buyer is computationally unbounded, the seller can signal his hidden information quite well, whereas this proves difficult if the buyer is computationally bounded.

6.1 DeMarzo's Model and the Liquidity Problem

First we sketch DeMarzo's ideas, but adapted to our setting. As in Section 2, we assume that the seller has N asset classes where seller holds C iid assets in each class. The number of lemon classes l is chosen at random from $[0, 2n]$. Assets in lemon classes will always default and assets in good classes will default with probability 1/2 independently. After normalization we also assume that each good asset class has expected payoff 1 dollar. As before the asymmetric information here is that only the seller knows which assets are lemons (and the number of lemons).

The seller wishes to sell assets for cash (e.g., because he has other investment opportunities). He has a discount factor $\delta < 1$: an asset that is worth 1 dollar contributes only δ to his utility, whereas 1 dollar in cash will contribute 1. Buyer's prior is that the number of lemon classes is uniform in $[0, 2n]$. Informally speaking, if the seller offers to sell all the assets, the buyers should interpret this as a "signal" that shows the seller's secret knowledge that the number of lemons is high, and will pay the minimum amount consistent with his prior (in our setting this is equal to the value of $N - 2n$ good asset classes). Clearly in this situation the seller cannot achieve a utility as high as when there's no asymmetric information. Thus the difference between seller's utility with and without asymmetric information is another way of measuring lemon cost. In [DeM05] and [DD99], DeMarzo and Duffie showed that this cost is minimized when the seller issues a CDO-like security and chooses the appropriate threshold to "signal" to the buyer about the quality of the asset portfolio.

To be more formal, let x be the expected total value of the assets (using seller's information), when the number of lemon classes is l. Thus $x = N - l$, which lies in the range $[N - 2n, N]$.

6.2 When seller sells a single security

This is the case studied in DeMarzo, which we summarize. In particular we will need his theorem below summarizing the liquidity cost in terms of the security design and the buyer's optimum response.

Suppose the seller offers to sell a fraction q of the security, and the buyer's evaluation for the entire pool of CDOs for this offer is $P(q)$. Then the seller's expected utility is

$$\delta(x - qf(x)) + P(q)q = \delta x + q(P(q) - \delta f(x)).$$

Given any function $P(q)$, the seller's optimum choice of q is given by function $Q^P(x) = \arg\max_q q(P(q) - \delta f(x))$ (here we ignored the first term δx because it is a constant given x).

The buyer's optimum response to the one-dimensional signal q must satisfy:

$$P(Q^P(x)) = \mathrm{E}[f(x)|Q^P(x)].$$

Theorem 7 (DeMarzo [DeM05], Page 11, Lemma 4). *If the structure of the security is fixed, and the range of f is between f_0 and f_1, then there is a unique separating equilibrium given by $Q(x) = (f(x)/f_0)^{-1/(1-\delta)}$, $P(q) = f_0 q^{\delta-1}$. The effect of asymmetric information (measured in seller's utility) is $(1-\delta)x - \pi(f(x)/f_0)f_0$, where $\pi(f(x)/f_0) = (1-\delta)(f(x)/f_0)^{-\delta/(1-\delta)}$.*

DeMarzo also proves that the optimum security is a *single* tranched CDO, but this result will not be of interest here. (Recall that we are interested in situations where the buyers has too many assets to realistically put them all in a single security.)

An example: We explain the above theorem with a calculation of the liquidity loss with some concrete numbers. Suppose $\delta = 0.9$, and and the ratio n/N is 0.1. Consider the utility loss of the buyer when $x = N - n$.
Case 1) *The seller doesn't use derivatives.* Then using $f(x) = x$ in DeMarzo's Theorem the utility loss is $(1-\delta)x - \pi(x/x_0)x_0 = (1-0.9)\cdot 0.9N - \pi(0.9/0.8)\cdot 0.8N \approx 6.23\%N$.
Case 2) *The seller uses the optimum security, a tranched CDO.* Suppose the seller creates a tranched CDO with the threshold of the CDO is set at $0.8N$. Then a simple calculation for this f shows that the utility loss is $(1-\delta)x - \pi(f(x)/f_0)f_0 = (1-0.9)\cdot 0.9N - \pi(0.8/0.8)\cdot 0.8N \approx 1\%N$, which is much smaller than the cost in Case 1. Thus the derivative does indeed reduce liquidity cost (which we called lemon cost).

6.3 When Seller sells multiple securities

Of course, the current paper is interested in the situation that the seller is not free to bundle all the assets into a single security and therefore needs to construct many securities. We assume (exogeneously) that the seller needs to construct M securities, each based upon D assets from different asset classes. The seller's ways to signal to the buyers is therefore limited to the structure of the CDO's and the fraction that the seller retains to show confidence. We further make the following assumption:

Symmetry Assumption: *The M CDOs have the same structure, and the seller offers to retain the same fraction for each of the CDO's he offers. That is, he does not split his offerings into different categories on this basis.*

The assumption is plausible from a practical standpoint as the buyers cannot distinguish these CDO's. We conjecture that this strategy is optimal for rational sellers though a proof seems difficult. We will provide some evidence for the truth of this conjecture below in Section 6.3.2.

For a subset of asset classes $S \subseteq [N]$ let $F(S)$ denotes the expected payoff of the portfolio of CDO's when exactly the asset classes in S are lemons. Define $f(x)$ be the worst case expected payoff of all the CDO's given the value of x, namely,

$$f(x) = \min_{|S|=N-x} F(S). \tag{2}$$

Clearly, a rational seller who knows the identity of the lemons would ensure such a payoff. Thus the following is a simple corollary of DeMarzo's analysis.

Theorem 8. *Theorem 7 holds for our setting with the f as defined in (2).*

6.3.1 Effect of computational complexity

Our Theorem 8 makes no reference thus far to computational complexity. Surprisingly, the main effect of computational complexity in this framework is to create doubt about the value of f_0, f_1 perceived by the buyer.

As in previous sections (specifically, Section 3, though similar results hold for other CDO's), here we assume (exogeneously) that the seller has created M binary CDO's using these N asset classes. Each CDO has D assets from different asset classes, and the threshold for a nonzero payoff is set to $(D - \Theta(\sqrt{D \log M}))/2$. What he controls is the choice of which assets to put in each CDO.
Case 1: The buyers are fully rational, and computationally unbounded. Then the asymmetric information is not a big problem. As indicated, the fully random CDO portfolio (constructed by randomly partitioning the mortgage bundle) can be examined by such a buyer (presumably using enormous computational power) to be free of booby traps. Essentially, the fully rational buyer will interpret the *lack of booby traps* of the CDOs constructed by the seller as a signal that shows the quality of the CDO bundle.

Thus, having verified the absence of a booby trap, the buyer will be able to correctly compute upper and lowerbounds (namely, f_1, f_0) on the function f. By results in Section 3 we know $f_0 \approx N - 2n$. (The true value of f_0 depends on D, the number of assets per CDO. When D grows $f_0/(N-2n)$ approaches 1 asymptotically and this value is close to 1 even for moderate D.) Furthermore, with extremely high probability $f_1 - f_0 = O(nN/\sqrt{D}M) \ll n$. When $M\sqrt{D}/N \gg 1$, the utility loss will be small. Concretely let's assume $f_0 = 0.75N$ and $f_1 - f_0 = 0.01N$, then the utitlity loss is roughly $2.34\%N$, although it is not optimal it is significantly lower than the loss than achieved without a derivative.

Case 2: Buyers are computationally limited. However, if the seller has no way to send a signal to the computationally limited buyer to prove there's no booby trap, then those buyers cannot properly interpret the above (high dimensional) signal as the fully rational buyers can.

The rational seller who has lots of lemon assets will always plant a booby trap because this will increase his utility and the sellers cannot notice. To avoid being cheated by the seller, the buyer will always assume there is a booby trap, and will conclude $f_1 - f_0 = \Theta(n\sqrt{N/M})$. Hence the utility loss for sellers who have just n lemon assets will be much larger. Concretely assume $f_0 = 0.6N$ and $f_1 = 0.8N$, the utility loss is roughly $8.54\%N$, which is even worse than what we had without a derivative. Also, notice the worst case of utility loss is $(1-\delta)x = 0.1 \cdot 0.9N = 9\%N$ because this is the utility loss if the seller does not sell anything.

6.3.2 Evidence for Symmetry Assumption

A priori it is unclear why the Symmetry Assumption is true: can a rational seller not try to signal his good intentions (and number of lemons) to the buyer by assigning different thresholds and fractions to each of the CDO? We argue that this is not likely to give the seller any profit by considering the following two examples:

Example 1 The seller breaks the assets into two portfolios in a bid to convey the following information to the buyers: " All lemons are in portfolio 1 and I'm giving it to you for free. Let's discuss what you will pay for portfolio 2, for which I offer to retain fraction q."

However, using the fact that the seller tries to optimize his utility, and the buyer tries to interpret the signals correctly, we can compute the signaling equilibrium in this situation using techniques similar to [DeM05] (Page 8, Lemma 2. An easy way to see this is by observing in Theorem 7, the values are only related to f_0, the lowerbound of f. Giving out some bad assets for free only changes the upperbound f_1 but not f_0.). The result shows that the seller's best strategy is to put no assets in portfolio 1. That is, this kind of signaling does not help the seller.

Example 2 The seller again breaks the assets into two portfolios, but this time the two portfolios has the same size. The seller try to retain different fractions for the two portfolios.

The idea of this example is maybe the seller can get profit by keeping different fraction of the CDOs to indicate which CDOs are better. However calculation shows that the best strategy for the seller is to keep the same fraction for the two portfolios.

Of course these two simple examples are suggestive but do not prove that the seller's optimal signaling mechanism is to offer to keep the same fraction for all the CDOs.

7 Design of Derivatives Resistant to Dense Subgraphs

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. We shall describe the construction in a future paper.

8 Conclusions

Most analysis of the current financial crisis blames the use of "faulty models" in pricing derivatives, and this analysis is probably correct. Coval et al. [CJS09] give a readable account of this "modelling error", and point out that buyer practices (and in particular the algorithms used by rating agencies [WA05]) 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 [Bru09].)

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 7 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.

Acknowledgements. We thank Moses Charikar for sharing with us the results from the manuscript [BCC$^+$09].

References

[ABW10] B. Applebaum, B. Barak, and A. Wigderson. Public-key cryptography from different assumptions. In *STOC*, pages 171–180, 2010.

[Ake70] G. Akerlof. The market for "lemons": quality uncertainty and the market mechanism. *The quarterly journal of economics*, pages 488–500, 1970.

[BC97] A. Bernardo and B. Cornell. The valuation of complex derivatives by major investment firms: Empirical evidence. *Journal of Finance*, pages 785–798, 1997.

[BCC$^+$09] 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.

[Bru09] M. Brunnermeier. Deciphering the liquidity and credit crunch 2007-08. *Journal of Economic Perspectives*, 23(1):77–100, 2009.

[CJS09] J. Coval, J. Jurek, and E. Stafford. The economics of structured finance. *Journal of Economic Perspectives*, 23(1):3–25, 2009.

[DD99] P. DeMarzo and D. Duffie. A liquidity-based model of security design. *Econometrica*, 67(1):65–100, 1999.

[DeM05] P. DeMarzo. The pooling and tranching of securities: A model of informed intermediation. *Review of Financial Studies*, 18(1):1–35, 2005.

[Duf07] D. Duffie. Innovations in credit risk transfer: implications for financial stability. *BIS Working Papers*, 2007.

[FPRS07] 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.

[GSe02] G. Gigerenzer, R. Selten, and eds. *Bounded rationality : the adaptive toolbox.* MIT Press, 2002.

[WA05] M. Whelton and M. Adelson. *CDOs-squared demystified.* Nomura Securities, February 2005.

A Our Construction versus Real-life Derivatives

In this section we discuss the relation of our constructions to commonly used types of derivatives on a more technical level.

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 unbeknownst 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. Though we did have results for lemon costs of tranched CDOs, the results for binary CDOs were stronger. This suggests firstly that binary CDOs may be a lot riskier than tranched CDOs, which was perhaps not quantified before in the same way. Second, the results for binary CDOs should still be important since they can be viewed as an abstraction for a debt insurance (i.e., CDS). If one buys a CDS (tranched) for a portfolio, then even though the tranching suggests a continuous cost incurred when the portfolio makes losses, in practice the cost is quite discontinuous as soon as the portfolio makes any loss, since this can cause rating downgrade (and subsequent huge drop in price), litigation, and other costs similar to that of a bankruptcy. Thus even a tranched CDS is a lot more binary in character in this setting.

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 E.5), 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 asset classes The distribution we used in our results, where every pair of assets is either independent (when they are in different asset classes) or perfectly correlated (when they are in the same asset class) is of course highly idealized. Also, the assumption that each asset is binary is only true in a highly simplified model. In practice the yields of assets might be assumed to come from some more realistic distribution, such as a gaussian copula. The Gaussian copula is constructed by projecting a multivariate normal distribution on $\mathbb{R}^d$ by means of the probability integral transform to the unit cube $[0,1]^d$. The Gaussian copula is capable of modeling various correlations between different assets. In particular, we can construct a gaussian copula $\mathcal{G}$ as follows.

Pick MD Gaussians that are partitioned into groups of size MD/N. The Gaussians within the same group are postively correlated and the Gaussians between different groups are independent. Let Σ be the correlation matrix for this multivariate Gaussian, the parameter Σ defines a gaussian copula

$$c_\Sigma(x) = \frac{1}{(\det \Sigma)^{1/2}} \exp\left\{-\frac{1}{2}u^T(\Sigma^{-1} - I)u\right\}.$$

Here $c_\Sigma(x)$ is the probability density that the yields of assets is equal to the vector x. The values of u will be determined as follows:

Now pick n groups as the junk asset classes, let $\mathcal{D}_{good}$ and $\mathcal{D}_{junk}$ be the distributions for yields of good and bad assets. Let F_{good} and F_{junk} be the corresponding cummulative distribution functions, then $u_i = \Phi^{-1}(F_{good}(x_i))$ if the i-th asset is good and $u_i = \Phi^{-1}(F_{junk}(x_i))$ otherwise. Assume $\mathrm{E}[\mathcal{D}_{good}] - \mathrm{E}[\mathcal{D}_{junk}] > c_1 \mathrm{E}[\mathcal{D}_{good}]$, higher moments of $\mathcal{D}_{good}$ and $\mathcal{D}_{junk}$ are bounded so concentration bounds hold. We have the following theorem:

Theorem 9. *There is a way for sellers to design derivatives such that the results summarized in Table 1 hold even when the yields of assets are chosen according to the Gaussian copula $\mathcal{G}$.*

The proof is essentially the same as before. Our proof does not depend crucially on the perfect correlation or the distribution of yield for each asset. We have mostly used concentration bounds. The proofs are correct even when the distribution of yields of individual assets is replaced by some more realistic distribution such as a Gaussian copula.

Another conern is that typically even for two dissimilar assets (from different asset classes), 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 even when the model has a systemwide correlation.

B Omitted proofs from Section 3

B.1 Maximizing Profit in Binary Derivative Setting

We provide a more complete analysis than the one in Section 3 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 $\frac{b}{2\delta} = a$, $x_i = t_i/2\delta$):

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

Here T is a parameter that is chosen to make sure computationally limited buyers cannot detect the dense subgraph. As argued in Section 3, 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.2 Omitted Proofs

Theorem 2 (Restated)**.** *When $n \ll md$, $\frac{dN}{Dn} > (N+M)^\epsilon$ for some constant ϵ, there is no dense subgraph (m,n,d) in a random (M,N,D) graph with high probability.*

Proof. Let $X_{i,j}$ be the random variable that indicates whether there's an edge between i-th asset class and j-th derivative. For simplicity we assume $X_{i,j}$ are independent random variables which has value 1 with probability D/N (actually the random variables $X_{i,j}$ are negatively correlated, but negative correlations will only make the results we are proving more likely to be true). Pick any set A of asset classes of size n, B of derivatives of size m. The probability that there are more than md edges between A and B is the probability of $X = \sum_{i\in A, j\in B} X_{i,j} \geq md$. Since $X_{i,j}$ are independent random variables, and the sum X has expectation $\mu = mnD/N$, by Chernoff Bound,

$$\Pr[X > (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$$

Here $\mu = mnD/N$, $1+\delta = md/\mu = \frac{dN}{Dn}$, the probability is at most

$$\Pr[X > (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \leq (N+M)^{-\epsilon md}$$

The number of such sets is $\binom{N}{n}\binom{M}{m} \leq (N+M)^{n+m}$, by union bound, the probability that there exists a pair of sets that has more than md edges between them is at most $(N+M)^{n+m-md}$, which is much smaller than 1 by the assumption that $n \ll md$. Therefore, with high probability there will be no dense subgraphs within a random graph. □

Theorem 3 (Restated)**.** *When $b > 3\sqrt{D}$, $d-b > 3\sqrt{D}$, $n/N \ll d/D$, a planted (m,n,d) subgraph will generate an extra lemon cost of at least $\Omega(mV)$. The Cost of Complexity for portfolios of binary derivatives is at least $\Omega(n\sqrt{N/M})$ under the densest subgraph assumption.*

Proof. It will be clear from the proof that $b = 3\sqrt{D}$, $d = 6\sqrt{D}$ is the hardest case. For concreteness we will assume $b = d-b = 3\sqrt{D}$ and other cases follow from the same arguments.

For each manipulated derivatives, let Y be the number of defaulted assets, since there are d assets that come from junk asset classes, 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, $\Pr[Y \geq \frac{D+b}{2}] = \Phi(3) = 1-p$ (here $p = \Phi(-3) \approx 0.2\%$ is a very small constant).

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

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

because the LHS and RHS are all expected number of defaulted assets. $x \geq \frac{D}{2} + \frac{2nD}{2N} - \frac{md}{2M-2m}$. The probability that the number of defaulted assets is more than $\frac{D+b}{2}$ is approximately $\Phi(-3+\frac{2n\sqrt{D}}{2N} - \frac{md}{2(M-m)\sqrt{D}}) \geq p - \Phi'(-3)\frac{md}{2(M-m)D} \approx p - \epsilon_1\frac{m}{M} + \epsilon_2 n\sqrt{D}/N$. Here ϵ_1 and ϵ_2 are all small constants. In conclusion, the expected number of derivatives that gives no return is roughly $pM+(1-p-\epsilon_1)m\epsilon_2 n\sqrt{D}/N$. When there are no lemons, this expectation is equal to pM. Therefore the lemon cost is $\epsilon_2 n\sqrt{D}/2+\Omega(mV)$. When the derivatives come from distribution $\mathcal{R}$, by Thm. 2 and Section B.1, the best strategy for the seller is to use a booby trap whose $m = O(n/d)$, and the lemon cost is small.

For Cost of Complexity, we consider distribution $\mathcal{R}$ and $\mathcal{P}$ in the densest subgraph assumption. The distribution $\mathcal{P}$ can have a dense subgraph with $m = n\sqrt{M/N}$. Since $\mathcal{P}$ and $\mathcal{R}$ are indistinguishable by the densest subgraph assumption, the cost of complexity is at least

$$CoC_\epsilon(\text{binary CDO}, n) \geq \Omega(n\sqrt{M/N}V - nV/\sqrt{D}) = \Omega(n\sqrt{N/M})$$

□

C Omitted proofs from Section 4

Theorem 4 (Restated). *When $b \geq 3\sqrt{D}$, $d-b \geq 3\sqrt{D}$, $d < \sqrt{D}\log D$, $n/N \ll d/D$, the lemon cost for a graph in distribution $\mathcal{P}$ (a graph with $(m, 2n, d)$ subgraph) is $\epsilon n + \tilde{\Theta}(\frac{\sigma}{D}mV)$, where $\epsilon = O(be^{-(b/2\sigma)^2/2}/\sigma)$. The Cost of Complexity of CDOs is at least $\Omega(n\sqrt{N/MD})$ under the densest subgraph assumption.*

Proof. Let Y_i be the number of defaulted assets related to i-th product.

If there are no junk assets, each asset may default with probability 1/2. For each product, Y_i is close to a Gaussian with standard deviation $\sigma = \sqrt{D}/2$ and mean $D/2$. The expected loss can be written as:

$$\frac{\sigma}{D}\cdot V \cdot \int_{-\infty}^{-3} \frac{-3-x}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\,dx$$

The value of the integral is a small constant $\epsilon = \frac{e^{-4.5}}{\sqrt{2\pi}} - 3\Phi(-3) < 10^{-3}$.

Valuation when there are n junk assets. When there are n junk assets, the expected value of each Y_i is decreased by $nD/2N$. Again, because $nD/2N$ is much smaller than 1, the amount of change in the expected payoff is proportional to $nD/2N\sigma$ and the derivative of the function $\Gamma(x) = \int_{-\infty}^{x} \frac{x-y}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}\,dy$ at the point $x = -b/2\sigma$. Since

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

$\Gamma'(-b/2\sigma) = O(b/2\sigma e^{-(b/2\sigma)^2/2})$. The lemon cost would be

$$M\frac{\sigma}{D}\cdot V\cdot\Gamma'(-b/2\sigma)\cdot\frac{nD}{2N\sigma} = O(n\cdot be^{-(b/2\sigma)^2/2}/\sigma).$$

Seller's profit from planting a dense subgraph. Suppose seller plants a dense subgraph (m, n, d). If the i-th product is in the dense subgraph, then Y_i is close to a Gaussian with mean $\frac{D+d}{2}$ and with high probability (the same probability $1-p = 99.8\%$) the senior tranche will suffer loss. Conditioned on the event that $Y_i > \frac{D+b}{2}$, the expected amount of loss is

$$\frac{V}{D}\cdot(\mathrm{E}[Y_i|Y_i > \frac{D+b}{2}] - \frac{D+b}{2}) \geq \frac{V}{D}\cdot(\mathrm{E}[Y_i] - \frac{D+b}{2}) = \frac{3\sigma}{D}V,$$

therefore the expected loss without conditioning is at least $\frac{3(1-p)\sigma}{D}V$. For the products outside the dense subgraph, because the expected value of Y_i is roughly $md/2M$ smaller (which is $md/2M\sigma$ times standard deviation), the expected loss is also smaller. The amount of change is the proportional to $md/2M\sigma$ and the derivative of the function $\Gamma(x)$ at the point $x = -3$. The total influence for all $M - m$ product is roughly $\frac{md}{2\sigma}\frac{\sigma}{D}V\Gamma'(-3) = \epsilon'\frac{\sigma}{D}mV$, where ϵ' is also a small constant, so this influence is very small.

For Cost of Complexity, again we use the distributions $\mathcal{R}$ and $\mathcal{P}$. The derivatives in $\mathcal{R}$ will have $m = n/\sqrt{D}$ with high probability, while the derivatives in $\mathcal{P}$ always have $m = n\sqrt{M/N}$. Thus the cost of complexity is at least

$$CoC_\epsilon(\text{CDO}, n) = \Omega(n\sqrt{M/N}V/\sqrt{D} - nV/D) = \Omega(n\sqrt{N/MD}).$$

□

D Omitted proofs from Section 5

Theorem 5 (Restated)**.** *In the binary CDO^2 setting, when $m^2 \gg \frac{M^2D^2}{N}$, $T = \frac{1}{2}(\mathrm{E}[random] + \mathrm{E}[planted])$, the lemon cost for derivatives in $\mathcal{P}$ is $(1-o(1))N/4$, while the lemon cost for derivatives in $\mathcal{R}$ is only $N/4e^{-m^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant. That is, the Cost of Complexity for binary CDO^{2}s is at least $\Omega(N)$.*

Proof. From the results in Section 3 we know $H = \mathrm{E}[\text{poly}]-\mathrm{E}[\exp] = \Theta(m)$. Therefore $\mathrm{E}[\text{poly}]-T = \Theta(m)$ and $T - \mathrm{E}[\exp] = \Theta(m)$.

Let Y be the random variable that's equal to the number of CDOs that gives no return, let X_i be the indicator variable that is 1 when i-th CDO gives no return and 0 otherwise. Then $Y = \sum_{i=1}^{M} X_i$. We analyse the concentration of Y about its expectation. Let $x_1, x_2, \ldots, x_N$ be the p value for asset classes, function $f((x)) = \mathrm{E}[Y|\mathbf{x}]$. Since after conditioning on $\mathbf{x}$, all assets are independent, and $H = \omega(\sqrt{M})$, $\Pr[|Y - \mathrm{E}[Y]| > H/4|\mathbf{x}] = e^{-\Theta(m^2/M)}$. Since each asset appears in $MD/N = N^{0.1}$ CDOs, $|f(\mathbf{x}) - f(\mathbf{x}')| \le MD/N$ if $\mathbf{x}$ and $\mathbf{x}'$ differ by only 1 position. By McDiarmid's bound:

$$\Pr[|f(\mathbf{x}) - \mathrm{E}[f(\mathbf{x})]| > t] \le e^{-\frac{t^2}{\sum c_i^2}}$$

where c_i is the maximal difference in f when the i-th component of $\mathbf{x}$ is changed. (here $c_i = MD/N$ for all i, so $\sum c_i^2 = M^2D^2/N$, for derivatives in distribution $\mathcal{P}$

$$\Pr[Y \ge T] \ge 1 - \Pr[f((x) \le T + H/4] - \Pr[|Y - \mathrm{E}[Y]| > H/4|\mathbf{x}] \ge\ge 1 - e^{-\Theta(m^2N/M^2D^2)},$$

similarly, for derivatives in distribution $\mathcal{R}$

$$\Pr[Y \ge T] \le \Pr[|Y - \mathrm{E}[\exp]| > H/4] + \Pr[|Y - \mathrm{E}[Y]| > H/4|\mathbf{x}] \le e^{-\Theta(m^2N/M^2D^2)}.$$

The lemon cost for this CDO^2 is just the probability of loss multiplied by the payoff of CDO^2. Therefore, exponential time buyer can distinguish $\mathcal{P}$ and $\mathcal{R}$, and make sure the lemon cost is $e^{-\Theta(m^2N/M^2D^2)}N/4$, which is $e^{-\Theta(N^{0.2})}$ in the parameters we choose. This is exponentially small, while polynomial buyers cannot distinguish $\mathcal{R}$ and $\mathcal{P}$ and may get lemon cost as large as $(1 - o(1))N/4$.

The cost of complexity is at least $\Omega(N)$ by using the distributions $\mathcal{R}$ and $\mathcal{P}$ in the definition.

□

Theorem 6 (Restated). *In the CDO^2 with tranching case, when $m^2d^2 \gg M^2D^2/N$, $T = \frac{1}{2}(\mathrm{E}[random] + \mathrm{E}[planted])$, the lemon cost for derivatives in distribution $\mathcal{P}$ is $\Theta(mdN/(MD))$, while the lemon cost for derivatives in distribution $\mathcal{R}$ is at most $mdN/MDe^{-m^2d^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant. That is, the Cost of Complexity for CDO^2's is at least $\Omega(n\sqrt{M/N})$ under the densest subgraph assumption.*

Proof. From Section 4 we know $H = \mathrm{E}[\text{poly}] - \mathrm{E}[\text{exp}] = \Theta(mdN/MD)$. Since $T = 1/2(\mathrm{E}[\text{poly}] + \mathrm{E}[\text{exp}])$, $\mathrm{E}[\text{poly}] - T = T - \mathrm{E}[\text{exp}] = \Theta(mdN/MD)$. Let X_i be amount of loss in the senior tranche of the i-th CDO, $Y = \sum_{i=1}^{M} X_i$, then the senior tranche of CDO^2 losses money if and only if $Y > T$.

Again, define $f(\mathbf{x}) = \mathrm{E}[Y|\mathbf{x}]$, where x_i's are p-values for asset classes. Still, after conditioning on $\mathbf{x}$, the assets are all independent so $\Pr[|Y - f((x))| > H/4|\mathbf{x}] = o(1)$. For the function $f(\mathbf{x})$, the c_i's for McDiarmid's bound are all 1, because the 1 unit of loss caused by the default of one asset can only be transferred and will not be amplified in CDOs with tranching. Therefore, for derivatives in distribution $\mathcal{P}$,

$$\Pr[Y > T] \geq 1 - \Pr[|Y - \mathrm{E}[\text{poly}]| > H/4] - \Pr[|Y - f((x))| > H/4|\mathbf{x}] \geq 1 - e^{-\Theta(m^2d^2N/M^2D^2)},$$

while for derivatives in distribution $\mathcal{R}$

$$\Pr[Y > T] \leq \Pr[|Y - \mathrm{E}[\text{exp}]| > H/4] + \Pr[|Y - f((x))| > H/4|\mathbf{x}] \leq e^{-\Theta(m^2d^2N/M^2D^2)}.$$

That is, polynomial time buyers may buy derivatives from distribution $\mathcal{P}$ and almost always loss money, their lemon cost is $(\mathrm{E}[\text{poly}] - T) = \Theta(mdN/MD)$, while the lemon cost for exponential time buyer is exponentially small because the probability of losing even \$1 is exponentially small (by the parameters we chose the probability is $e^{-\Theta(N^{0.5})}$). Again we use the distributions $\mathcal{P}$ and $\mathcal{R}$ in the definition of cost of complexity, and conclude that $CoC_\epsilon(\mathrm{CDO}^2, n) = \Omega(n\sqrt{N/MD})$. □

E Survey of Algorithms for Dense Subgraph

Since designers of financial derivatives can use *booby traps* 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 [WA05]. There is some literature on how to design derivatives, specifically, using ideas such as *combinatorial designs* [FPRS07]. 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. We shall see that there is a wide range of parameters (see Section E.6) where dense subgraphs are not detectable by any known algorithms. Further, detecting dense subgraphs under these parameters are assumed to be hard for *every* efficient algorithm.The dense subgraph parameters used in our constructions all fall in 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. Also, several algorithms (especially the Semi-definite Programming in Section E.3) we described requires more than linear or quadratic time. Although they still runs in polynomial time in practice they are hard to implement for large-scale data.

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.

E.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.

E.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 α be a length 4 cycle, then X_α 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 α and β 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 α 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.

E.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 [BCC+09], 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.

E.4 Log Density

In [BCC+09] 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 $\log d/\log n < \log D/\log N$.

E.5 Ex-Ante and Ex-Post Detection

So far in this paper we were concerned with detecting 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. But one 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. asset classes bipartite graphs, and this can happen after all assets results are known— we call this *ex post detection* of dense subgraphs, contrasted with the normal, sell-time *ex ante detection* that is our focus in most of this paper.[9]

[9]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

Ex-Post detection is easier than Ex-Ante detection, because we have more information after the result has been revealed. We shall change our model slightly. For each asset class we have a probability p_{class}, which is the probability of default in this class. At first this p_{class} is drawn according to some distribution, then conditioned on this probability, all assets in this class will default with probability p_{class}. The distribution of p_{class} in good asset classes and junk asset classes are different.

In some situations, the distribution of p_{class} will reveal a lot of information for Ex-Post detection. The easiest choice $p_{class} = 1/2$ for good classes and $p_{class} = 1$ for junk classes leads to Ex-Post detectability, because the assets in junk classes all defaulted, but for a good class, only with exponentially low probability all assets will default, after we learned whether the assets has defaulted or not, we would be able to tell which classes are lemons and verify the existence of the dense subgraph. Also, when $b = c\sqrt{D \log D}$ for some large enough c, 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. One observation here is: when the expected number of junk asset classes or derivatives with no return is small in a random graph, almost all vertices with unusual behavior (such as a derivative that defaults or an asset class with much more defaults than average) will be in the planted dense subgraph, which makes detecting dense subgraph easy. But when the number of derivatives with no return is large (e.g. $\Omega(M)$), and the distributions of p_{class} is carefully chosen, then it still seems hard to find the dense subgraph even ex-post. To rule out trivial methods for ex-post detection, the distributions of p_{class} in good and junk classes should not differ by too much. For example, p_{class} is uniformly chosen from $[0.2, 0.8]$ for good classes, while for junk classes p_{class} is uniformly chosen from $[0.5, 0.8]$. On average, an asset from junk classes still have significantly higher probability of default than an asset from good classes (0.65 vs. 0.5), which enables dense subgraph to manipulate the result. At the same time, half of the good classes will have p_{class} from $[0.5, 0.8]$ and look indistinguishable with junk classes, so junk classes still hide among the good classes even after results are revealed. We do not have a proof based on Densest Subgraph Assumption that shows it is impossible to find dense subgraph in this Ex-Post setting, but can show a relation between finding the planted subgraph in the ex-post and ex-ante setting in a slightly different 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 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 10. *Suppose that the added dense subgraph search problem is hard for parameters*

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".

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".

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. □

Combining the observation about p_{class} and Theorem 10, we have:

Theorem 11. *Suppose the asset classes satisfy the model in this Section, where p_{class} is distributed according to $\mathcal{D}_{good}$ for good asset classes and $\mathcal{D}_{junk}$ for junk classes. Let d_{good} and d_{junk} be the density function of $\mathcal{D}_{good}$ and $\mathcal{D}_{junk}$. If $d_{good}(x) \geq 1/2 d_{junk}(x)$ for all $x \in [0,1]$, the expectation of $\mathcal{D}_{good}$ and $\mathcal{D}_{junk}$ differ by at least a constant, and the graph is generated according to Theorem 10 with appropriate parameters M, N, D, m, n, d such that the added dense subgraph search problem is hard, then there is a way to design derivatives so that the lemon cost is larger (as in first column of Table 2) for booby-trapped graphs and the buyers cannot find the dense subgraph even ex post.*

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 claim that the buyers cannot distinguish the two distributions. However the buyers won't be able to find out where the booby trap is.One particular problem about the search problem is we can no longer set $b = \sqrt{D \log D}$. Instead we should take $b = O(\sqrt{D})$ so that the problem remains hard. 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. Table 2 contains our result when the threshold b is constrained to be $O(\sqrt{D})$, in which case it's plausible that ex-post undetectability holds, as well (for comparison's sake) the corresponding results for our usual setting of $b = \sqrt{D \log D}$, which is generally ex-post detectable by the algorithm of just looking at the derivatives of 0 value.

Model	**Lemon cost for** $b = 3\sqrt{D}$	**Lemon cost for** $b = O(\sqrt{D \log D})$
Derivative-free	n	n
binary CDO, exp time	$\epsilon n\sqrt{D} + n(N/M\sqrt{D})$	$\sim n(N/M\sqrt{D}) \ll n$
binary CDO, poly time	$\epsilon n\sqrt{D} + n\sqrt{N/M}$	$\sim n\sqrt{N/M} \gg n$
tranched CDO, exp time	$\epsilon n + n(N/MD)$	$n(N/MD)$
tranched CDO, poly time	$\epsilon n + n(\sqrt{N/MD})$	$n(\sqrt{N/MD})$
binary CDO^2, exp time	0	0
binary CDO^2, poly time	$N/4$	$N/4$
tranched CDO^2, exp time	0	0
tranched CDO^2, poly time	$n(\sqrt{N/MD})$	$n(\sqrt{N/MD})$

Table 2: Lemon cost results based on value of threshold b. While in our parameters derivatives with $b \sim \sqrt{D \log D}$ are easily ex-post detectable, it is not clear whether this holds for $b = O(\sqrt{D})$.

E.6 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. All the parameters we mentioned in previous Sections satisfy these constraints.

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 Theorem 2. 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 E.2) and SDP (Section E.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 derivatives will be smaller than n, while the lemon cost for polynomial time buyers is always larger than n. See Section 3

8. $m^2 \gg M^2D^2/N$ or $m^2d^2 \gg M^2D^2/N$

   When the former is satisfied, then binary CDO^{2}s can have exponential difference between the lemon cost for exponential time buyer and polynomial time buyer. When the latter is satisfied, then CDO^{2}s with tranching can have exponential difference between the lemon cost for exponential time buyer and polynomial time buyer. See Theorem 5 and Theorem 6

9. $\log d/\log m < \log D/\log M$ 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 [BCC$^+$09]) can detect the dense subgraph.