Computational Complexity and Information Asymmetry in Financial Products

(Working paper)

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

October 19, 2009

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.

See also the webpage `http://www.cs.princeton.edu/~rongge/derivativeFAQ.html` for an informal discussion on the relevance of this paper to derivative pricing in practice.

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

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

1 Introduction

A *financial derivative* is a contract entered between two parties, in which they agree to exchange payments based on the performance or events relating to one or more underlying assets. The securitization of cash flows using financial derivatives transformed the financial industry over the last three decades. In recent years derivatives have grown tremendously, both in market volume and sophistication. The total volume of trades dwarfs the world's GDP. This growth has attracted criticism (Warren Buffet famously called derivatives "financial weapons of mass destruction"), and many believe derivatives played a role in enabling problems in a relatively small market (U.S. subprime lending) to cause a global recession. (See Appendix A for more background on financial derivatives and [CJS09, Bru08] for a survey of the role played by derivatives in the recent crash.)

Critics have suggested that derivatives should be regulated by a federal agency similar to FDA or USDA. Opponents of regulation counter that derivatives are contracts entered into by sophisticated investors in a free market, and play an important beneficial role. All this would be greatly harmed by a slow-moving regulatory regime akin to that for medicinal drugs and food products.

From a theoretical viewpoint, derivatives can be beneficial by "completing markets" 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 less information-sensitive part of the cash flow stream of an asset (e.g., a mortgage) and retain the remainder. DeMarzo [DeM05] suggests this beneficial effect is quite large. (We refer economist readers to Section 1.3 for a comparison of our work with DeMarzo's.)

The practical downside of using derivatives is that they are *complex* assets that are difficult to price. Since their values depend on complex interaction of numerous attributes, the issuer can easily tamper derivatives without anybody being able to detect it within a reasonable amount of time. Studies suggest that valuations for a given 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]. Many sources for this complexity have been identified, including the complicated structure of many derivatives, the sheer volume of financial transactions, the need for highly precise economic modeling, the lack of transparency in the markets, and more. Recent regulatory proposals focus on improving transparency as a partial solution to the complexity.

This paper puts forward *computational complexity* as an aspect that is largely ignored in such discussions. One of our main results suggests that it may be computationally intractable to price derivatives even when buyers know almost all of the relevant information, and furthermore this is true even in very simple models of asset yields. (Note that since this is a hardness result, if it holds in simpler models it also extends for any more model that contains the simpler model as a subcase.) This result immediately posts a red flag about asymmetric information, since it implies that derivative contracts could contain information that is in plain view yet cannot be understood with any foreseeable amount of computational effort. This can be viewed as an extreme case of bounded rationality [GSe02] whereby even the most sophisticated investment banks such as Goldman Sachs cannot be fully rational since they do not have unbounded computational power. We show that designers of financial products can rely on computational intractability to disguise their information via suitable "cherry picking." They can generate extra profits from this *hidden information*, far beyond what would be possible in a fully rational setting. This suggests a revision of the accepted view about the power of derivatives to ameliorate the effects of information asymmetry.

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

Computational complexity and informational asymmetry Computational complexity studies *intractable* problems, those that require more computational resources than can be provided by the fastest computers on earth put together. 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 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,[1] 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.[2] 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.

For more information about computational complexity and intractability, we refer readers to the book by Arora and Barak [AB09].

Akerloff's notion of lemon costs and connection to intractabilty. Akerloff's classic 1970 paper [Ake70] gives us a simple framework for quantifying asymmetric information. The simplest setting is as follows. You are in the market for a used car. A used car in working condition is worth \$1000. However, 20% of the used cars are *lemons* (i.e., are useless, even though they look fine on the outside) and their true worth is \$0. Thus if you could pick a used car at random then its expected worth would be only \$800 and not \$1000. Now consider the seller's perspective. Suppose sellers know whether or not they have a lemon or not. Then a seller who knows that his car is not a lemon would be unwilling to sell for \$800, and would exit the market. Thus the market would feature only lemons, and nobody would buy or sell. Akerloff's paper goes on to analyze reasons why used cars do sell in real life. We will be interested in one of the reasons, namely, that there could be a *difference* between what a car is worth to a buyer versus a seller. In the above example, the seller's value for a working car will have to be \$200 less than the buyer's in order for trade to occur. In this case we say that the "lemon cost" of this market is \$200. Generally, the higher this cost, the less efficient is the market.

We will measure the cost of complexity by comparing the lemon cost in the case that the buyers are computationally unbounded, and the case that they can only do polynomial-time computation. We will show that there is a significant difference between the two scenarios.

Results of this paper. From a distance, our results should not look surprising to computer scientists. Consider for example a derivative whose contract contains a 1000 digit integer n and has

[1] 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.

[2] 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.

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.

The high level idea is that these everyday derivatives are based on applying threshold functions on various subsets of the asset pools. Our result is related to the well known fact that random election involving n voters can be swung with significant probability by making $\sqrt{n}$ voters vote the same way. *Private information* for the seller can be viewed as a restriction of the input distribution known only to the seller. The seller can structure the derivative so that this private information corresponds to "swinging the election." What is surprising is that a computationally limited buyer may not have any way to distinguish such a *tampered* derivative from untampered derivatives. Formally, the indistinguishability relies upon the conjectured intractability of the *planted dense subgraph* problem.[3] This is a well studied problem in combinatorial optimization (e.g., see [FPK01, Kho04, BCC$^+$09]), and the planted variant of it has also been recently proposed by Appelbaum et al. [ABW09] as a basis for a public-key cryptosystem.

Note that the lemon issue for derivatives has been examined before. It is well-recognized that since a seller is more knowledgeable about the assets he is selling, he may design the derivative advantageously for himself by suitable cherry-picking. However since securitization with derivatives usually involves *tranching* (see Section 3 and Appendix A), and the seller retains the junior tranche which takes the first losses, it was felt that this is sufficient deterrence against cherry-picking (ignoring for now the issue of how the seller can be restrained from later selling the junior tranche). We will show below that this assumption is incorrect in our setting, and even tranching is no safeguard against cherry-picking.

Would a lemons law for derivatives (or an equivalent in terms of a standard clauses in CDO contracts) remove the problems identified in this paper? The paper suggests (see Section F.4 in the appendix.)) a surprising answer: in many models, even the problem of detecting the tampering *ex post* may be intractable. The paper also contains some results at the end (see Section 5) that suggest that the problems identified in this paper could be mitigated to a great extent by using certain exotic derivatives whose design (and pricing) is influenced by computer science ideas. Though these are provably tamper proof in our simpler model, it remains to be seen if they can find economic utility in more realistic settings.

1.1 An illustrative example

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

[3] Note that debt-rating agencies such as Moody's or S&P currently use simple simulation-based approaches [WA05], and hence may fail to detect tampering even in the parameter regime where the densest subgraph is easy.

say that the "lemon cost" in this setting is $n/2$.

In principle one can use derivatives to significantly ameliorate the lemon cost. In particular consider the following: seller creates M new financial products, each of them depending on D of the underlying assets.[4] Each one of the M products pays off $N/(3M)$ units as long as the number of assets in its pool that defaulted is at most $D/2 + t\sqrt{D}$ for some parameter t (set to be about $\sqrt{\log D}$), and otherwise it pays 0. Henceforth we call such a product a "Binary CDO".[5] Thus, if there are no lemons then the combined value of these M products, denoted V, is very close to $N/3$.

One can check (see Section 2) that if the pooling is done randomly (each product depends on D random assets), then even if there are n lemons, the value is still $V - o(n)$, no matter where these lemons are. We see that in this case derivatives do indeed help significantly reduce the lemon cost from n to $o(n)$, thus performing their task of allowing a party to sell off the least information-sensitive portion of the risk.

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

Clearly, a fully rational (i.e., computationally unbounded) buyer can enumerate over all possible n-sized subsets of $[N]$ and verify that none of them are over-represented, thus ensuring a lemon cost of $o(n)$. However, for a real-life buyer who is computationally bounded, this enumeration is infeasible. In fact, the problem of detecting such a tampering is equivalent to the so-called *hidden dense subgraph* problem, which computer scientists believe to be intractable (see discussion below in Section 1.2). Moreover, under seemingly reasonable assumptions, there is a way for the seller to "plant" a set S of such over-represented assets in a way that the resulting pooling will be *computationally indistinguishable* from a random pooling. The bottom line is that under computational assumptions, the lemon cost for polynomial time buyers can be much larger than n. Thus introducing derivatives into the picture *amplifies* the lemon cost instead of reducing it!

Can the cost of complexity be mitigated? In Akerloff's classic analysis, the no-trade outcome dictated by lemon costs can be mitigated by appropriate signalling mechanism —e.g., car dealers offering warranties to increase confidence that the car being sold is not a lemon. In the above setting however, there seems to be no direct way for seller to prove that the financial product is *untampered*. (It is believed that there is no simple way to prove the absence of a dense subgraph; this is related to the $NP \neq coNP$ conjecture.). Furthermore, we can show that for suitable parameter choices the tampering is *undetectable* by the buyer even *ex post*. The buyer realizes at the end that the financial products had a higher default rate than expected, but would be unable to prove that this was due to the seller's tampering. (See Section F.4 in the appendix.) Nevertheless, we do show in Section 5 that one could use Computer Science ideas in designing derivatives that are tamperproof in our simple setting.

[4] We will have $MD \gg N$ and so the same asset will be contained in more than one product. In modern finance this is normal since different derivatives can reference the same asset. But that one can also think of this overlap between products as occurring from having products that contain assets that are strongly correlated (e.g., mortgages from the same segment of the market). Note that while our model may look simple, similar results can be proven in other models where there are dependencies between different assets (e.g., the industry standard Gaussian copula model), see discussion in Section A.1.

[5] 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.1.

1.2 The cost of complexity: definitions and results

We now turn to formally defining the concept of "lemon cost", and stating our results. For any derivative F on N inputs, input distribution X over $\{0,1\}^N$, and $n \leq N$, we define the *lemon cost* of F for n junk assets as

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

where the min takes into account all possible ways in which seller could "position" the junk assets among the N assets while designing the derivative. (We'll often drop the subscripts F, X when they are clear from context.) The lemon cost captures the inefficiency introduced in the market due to the existence of "lemons" or junk assets. For example, in the Akerlof setting, if half the sellers are honest and have no junk assets, while half of them have n junk assets which they naturally place in the way that minimize the yield of the derivative, then a buyer will pay $V - \Delta(n)/2$ for the derivative, where V is the value in the junk-free case.[6] Hence the buyer's valuation will have to be roughly $\Delta(n)$ above the seller's for trade to occur.

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 also list the lemon cost for buyers with infinite computational power ("fully rational"). 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 4) this gap can be much larger with essentially zero lemon cost in the exponential case and maximal cost in the polynomial case.

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

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 $\mathcal{D}$ on (M, N, D) graphs:

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

[6] A similar observation holds if we assume the number of junk assets is chosen at random between 0 and n.

Model	Lemon cost	Reference
Derivative-free	n	
binary CDO, exp time	$\sim n(N/M\sqrt{D}) \ll n$	Theorem 2
binary CDO, poly time	$\sim n\sqrt{N/M} \gg n$	Theorem 2
tranched CDO, exp time	$\sim n(N/MD)$	Theorem 3
tranched CDO, poly time	$\sim n(\sqrt{N/MD})$	Theorem 3
binary CDO^2, exp time	0	Theorem 4
binary CDO^2, poly time	$N/4$	Theorem 4
tranched CDO^2, exp time	0	Theorem 5
tranched CDO^2, poly time	$\sim n(\sqrt{N/MD})$	Theorem 5

Table 1: Cost of n junk assets 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.

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 [ABW09] as a source for public key cryptography[7]. 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 $N = o(MD)$, $(md^2/n)^2 = o(MD^2/N)$ 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 $m = \tilde{O}(n\sqrt{M/N})$.

1.3 Comparison with DeMarzo(2005)

DeMarzo (2005) (and earlier, DeMarzo and Duffie (1999)) considers a simple model of how CDOs can help ameliorate asymmetric information effects. Since we show that this conclusion does not always hold, it may be useful to understand the differences between the two approaches. The full version of this paper will contain an expanded version of this section.

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. 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 the one of the potential 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

[7] 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.

(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 tranch, 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 and the seller is not at a disadvantage because the lemon costs go to 0 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 t iid assets in each class. 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. Second, 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.

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. Seller randomly distributes his assets into M equal sized pools and defines the senior tranche identically in all of them. Thus his "signal" to buyers consists of the partition of assets into pools, and the threshold that defines the senior tranche. For a fully rational buyer this signal turns out to contain enough information: he only needs to verify that there is no dense subgraph in the graph that defines the CDOs. If a dense subgraph is found, the buyers can assume that the seller is trying to gain advantage by clustering the lemons into a small number of CDOs (as in our construction), and will lower their offer price for the senior tranche accordingly. If no dense subgraph is found then buyers can have confidence that the small number of lemons are more or less uniformly distributed among the M tranches, and thus have only a negligible effect on the senior tranche. Assuming the number of lemons is not too large, the lemon costs go to 0 as $N \to \infty$, confirming DeMarzo's findings in this case. Thus lack of a dense subgraph is a "signal" from seller to buyer that there is no reason to worry about the lemons in the CDOs.

Of course, detecting this "signal" is computationally difficult! If buyers are computationally bounded then this signalling cannot work, and indeed our assumption about the densest subgraph problem implies that the buyers cannot detect the presence of a dense subgraph (or its absence) with any reasonable confidence. Thus lemon costs persist. (A more formal statement of this result involves showing that for every possible polynomial-time function that represents the buyer's "interpretation" of the signal, the seller has a strategy of confusing it by planting a dense subgraph.)

Ex post indetectability At the superficial level described above, the seller's tampering is detectable ex post. In Section F.4.1 in the appendix we describe how a small change to the above model makes the tampering undetectable ex post.

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

Since each derivative depends on D independent assets, the number of defaulted assets for each derivative is distributed very closely to a Gaussian distribution as D gets larger. In particular, if there are no lemons, every derivative has exactly the same probability of paying off, and this probability (which as b grows becomes very close to 1) is closely approximated by $\Phi(\frac{b}{2\sigma})$ where 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 $N\frac{D-b}{2D}\frac{N}{M} \sim N/2$. Note that this calculation is the same regardless of whether the graph is random or not.

We now compute the effect of n lemons (i.e., assets with payoff identical to 0) on the value of all the derivatives. In this case the shape of the pooling *will* make a difference. It is convenient to view the relationship of derivatives and assets as a *bipartite graph*, see Figure 1. Derivatives and assets are vertices, with an edge between a derivative and an asset if the derivative depends on this asset. Note that this is what we called an (M, N, D) graph.

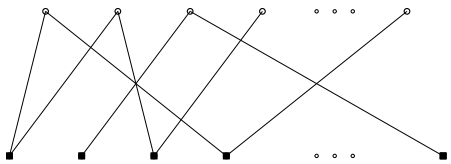

Figure 1: Using a bipartite Graph to represent assets and derivatives. There are M vertices on top corresponding to the derivatives and N vertices at the bottom corresponding to assets. Each derivative references D assets.

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 affect its payoff probability it suffices to fix about $\sigma \sim \sqrt{D}$ of the underlying assets. More precisely, if t of the assets contained in a derivative are lemons, then the expected number of defaulted assets in it is $\frac{D+t}{2}$, while the standard deviation is $\sqrt{D-t}/2 \approx \sqrt{D}/2$. Hence the probability that this derivative gives 0 return is $\Phi(\frac{t-b}{2\sigma})$ which starts getting larger as t increases. This means that the difference in value between such a pool and a pool without lemons is about $V \cdot \Phi(\frac{t-b}{2\sigma})$.

Suppose the seller allocates t_i of the junk assets to the ith derivative. Since each of the n junk assets are contained in MD/N derivatives, we have $\sum_{i=1}^{M} t_i = \frac{nMD}{N}$. In this case the lemon cost will be

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

Since the function $\Phi(\frac{t_i-b}{2\sigma})$ is concave when $t < b$, and convex after that the optimum solution will involve all t_i-s to be either 0 or $k\sqrt{D}$ for some small constant k. (There is no closed form for k but it is easily calculated numerically; see Section B.1.)

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

Note that even random graphs have dense subgraphs. For example, when $md = n$, any graph has an (m, n, d) subgraph— just take any m top vertices and their $n = md$ neighbors. But these are more or less the densest subgraphs in random graphs, as the following theorem, proven in Section B, shows:

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

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

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, n, 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 ensure a lemon cost of $n\frac{N}{2M\sqrt{D}} = o(n)$, while a polynomial time buyer will have lemon cost of $n\sqrt{N/M} = \omega(n)$.

3 Non-Binary Derivatives and Tranching

We showed in Section 2 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 absorb all losses until it's "wiped out" in which case losses start propagating up to the more senior tranches. Thus the most senior tranche is considered very safe, and often receives 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 only people who owns junior tranche suffers the loss; if the loss is above the threshold, then junior tranche lose all its value and senior

tranche is responsible for the rest of the loss. Clearly, senior tranches have lower risk than the whole asset. We will assume that seller retains the junior tranche, which is normally believed to signal his belief in the quality of the asset. Intuitively, such tranching should make the derivative 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 2 —that the lemon cost is much larger when we take computational complexity into account—is unchanged.

We adapt the parameters and settings in Section 2, the only difference is that we replace our binary derivative with the senior tranche derivative, that in the event of $T > \frac{D+b}{2}$ defaults, does not have a payoff of 0, but rather a payoff of $D - T$ (and otherwise has a payoff of $V = \frac{D-b}{2}$ 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 changing any assignment to the assets by making one asset default can cause at most a total loss of one unit to the sum of all derivatives. In fact we will show that in this case for both the exponential-time and polynomial-time buyer, the lemon cost is *less* than n. But there will still be a difference between the two case, specifically we'll have that the cost is δn in the exponential-time case and $\sqrt{\delta} n$ in the polynomial-time case, where $\delta \sim \frac{N}{MD}$, as is shown by the following theorem (whose proof is deferred to Appendix C):

Theorem 3. *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 with an (m, n, d) subgraph is $\epsilon n + \tilde{\Theta}(\frac{\sigma}{D} mV)$, where $\epsilon = O(be^{-(b/2\sigma)^2/2}/\sigma)$.*

Therefore in CDOs, setting b sufficiently large the lemon cost for polynomial time buyers is $\Theta(n\sqrt{N/MD})$, while 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.

4 Lemon Cost for CDO^2

Though CDOs are the most common derivatives, there is still significant trade in a more complex derivative called CDO^2, which is a CDO of CDOs. (Indeed, there is trade even in CDO^3, which are CDOs of CDO^2, and more generally there are CDO^n!) In this section we'll examine CDO^2's using lemon cost and computational complexity, and show that it is more vulnerable to dense subgraphs than the simple CDO. The effect is more pronounced in CDO^2 that is a binary CDO of binary CDOs considered in Section 2. We defer the proofs of the theorems below to Appendix D.

For this section to simplify parameters we set $M = N^{0.8}$, $D = N^{0.3}$, $n = N^{0.8}$, $m = N^{0.7}$, $d = 6\sqrt{D}$. For the first level of CDOs we put the threshold at $\frac{D+b}{2}$ defaults where $b = 3\sqrt{D}$. One can easily check that these parameters satisfy all the requirements for any dense subgraphs to escape detection 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 2, and combines them to make a single derivative. The CDO^2 gives return $V = N/4$ when the number of binary CDOs that gives no return is more than a threshold T (which we will set later, T will be small so that the CDO^2 is securitized and can be paid for out of the seller's profits from the N assets) and gives no return otherwise. From Section 2 we know when buyer has unbounded computational power and consequently there are no "booby traps", the expected number of CDOs that gives no return is roughly $pM + \epsilon n\sqrt{D} + nM\sqrt{D}/N$ where $p = \Phi(-3) < 0.2\%$; while for polynomial time buyer, the expected number of CDOs that

gives no return can be as large as $pM + \epsilon n\sqrt{D} + n\sqrt{M/N}$. We denote the former as E[exp] and the latter as E[poly].

Theorem 4. *In the binary CDO^2 setting, when $m^2 \gg \frac{M^2D^2}{N}$, $T = \frac{1}{2}(\mathrm{E}[poly] + \mathrm{E}[exp])$, the lemon cost for polynomial time buyer is $(1 - o(1))N/4$, while the lemon cost for exponential time buyer is only $N/4e^{-m^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant.*

Now we consider the more popular CDO^{2}s, which uses tranching. At the first level we use the same way of dividing the CDOs into two tranches as in Section 3. Then we collect all the senior tranche of the CDOs to make a CDO^2. Since each senior tranche of CDO has value $\frac{D-b}{2D}\frac{N}{M}$, the total value of the CDO^2 is $V = N\frac{D-b}{2D}$. The CDO^2 is also splitted into two tranches, and the threshold T is determined later. The buyers buy the senior tranche of CDO^2 and seller retains the junior tranche.

Use E[exp] to denote the expected loss of CDO^2 for exponential time buyers, E[poly] to denote the expected loss of CDO^2 for polynomial time buyers, we have the following theorem.

Theorem 5. *In the CDO^2 with tranching case, when $m^2d^2 \gg M^2D^2/N$, $T = \frac{1}{2}(\mathrm{E}[poly] + \mathrm{E}[exp])$, the lemon cost for polynomial time buyers is $\Theta(mdN/MD)$, while the lemon cost for exponential time buyer is at most $mdN/MDe^{-m^2d^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant.*

In conclusion, for both binary CDO^{2}s and CDO^{2}s with tranching, the lemon cost for polynomial time buyers can be exponential times larger than the lemon cost for exponential time buyers. In particular, the lemon cost of binary CDO^{2}s for polynomial time buyers can be as large as $N/4$— a large fraction of the total value of the asset.

5 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 briefly outline the constructions — details can be found in Appendix E.

The XOR function. The XOR function on D variables is of course very resilient to any setting of much less than D variables (even if the variables are not fair but rather biased coins). Hence we can show that an XOR based derivative will have $o(n)$ lemon cost even for computational buyers. However, it's not clear that this function will be of economic use, as it's not monotone, and also could be sucseptible to a timing attack, in which the seller will manipulate the last asset after the outcome all other ones is known.

Tree of majorities. Another, perhaps more plausible candidate, is obtained by applying majorities of 3 variables recursively for depth k, to obtain a function on 3^k variables that cannot be influenced by $o(2^k) \gg \sqrt{3^k}$ variables. In this case we can show that the added resiliency of the function allows to detect dense subgraph *ex-post*— after the outcome of all assets is known. This is a non-trivial and possibly useful property (see discussion in Section F.4) as one may be able to add to the contract a clause saying that the detection of a dense subgraph *whose input variables mostly ended up yielding* 0 incurs a sizeable payment from seller to buyer. Alternatively, this "penalty" may involve possibility of law enforcement or litigation.

6 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 [Bru08].)

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 5 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

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[ABW09] B. Applebaum, B. Barak, and A. Wigderson. Public key cryptography from different assumptions. Available from the authors' web pages. Preliminary version as cryptology eprint report 2008/335 by Barak and Wigderson., 2009.

[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.

[Bru08] M. Brunnermeier. Deciphering the 2007-08 liquidity and credit crunch. *Journal of Economic Perspectives*, 23(1):77–100, 2008.

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

[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.

[FPK01] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[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.

[Kho04] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *FOCS*, pages 136–145, 2004.

[McD03] R. McDonald. *Derivatives markets*. Addison-Wesley Reading, MA, 2003.

[Mou08] C. Mounfield. *Synthetic CDOs: Modelling, Valuation and Risk Management*. Cambridge Univ Pr, 2008.

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

A Real world financial derivatives

In this appendix we provide a quick review of financial derivatives. See the book [McD03] for a general treatment and [Mou08] for a treatment of the practical aspects of evaluating the kind of derivatives discussed here. The papers [CJS09, Bru08] discuss the role derivatives played in bringing about the recent economic crisis. In Section A.1 we compare the features of our model with the kind of derivatives used in practice.

A derivative is just a contract entered between two parties to exchange payments based on some events to underlying assets. Derivatives are extremely general: the events can be discrete events such as default or credit rating change, as well as just market performance. The assets themselves can be a company, a loan, or even another derivative. Neither party has to actually own the assets in question. The payment structure can also be fairly complex, and can also depend on the timing of the event.

A relatively simple example of a derivative is a *credit default swap (CDS)*, in which Party A "insures" Party B against default of a third Party X. Thus an abstraction of a CDS (ignoring timing of defaults) is that if we let X be a random variable equalling 0 if Party X defaults (say in one year) and equal to 1 otherwise, then Party A pays a certain amount iff $X = 0$. Naturally, the value that B will pay A for this contract will be proportional to the estimated probability that X defaults. We note that despite the name "insurance", there is no requirement that Party B holds a bond of Party X, and so it is possible for Party B to "insure" an asset it does not own, and there valid economic reasons why it would want to do so. Indeed there are entities X for which the volume of CDS derivatives that reference X far exceeds the total of all X's assets and outstanding debts.

A *collateralized debt obligation (CDO)*,[8] can be thought of as a generalization of CDS that references many, say D parties X_1,...,X_D. Again we say that X_i is equal to 0 if Party X_i defaults, and to 0 otherwise, and this time Party B pays Party A $F_{\alpha,\beta}(X_1, \ldots, X_D)$ units, for some $0 \leq \alpha < \beta \leq 1$, where $F_{\alpha,\beta}$ is defined as follows:

$$F_{\alpha,\beta}(X_1, \ldots, X_D) = \begin{cases} 0 & S \leq \alpha D \\ 1 & S \geq \beta D \\ (S - \alpha D)/(\beta - \alpha) & S \in (\alpha D, \beta D) \end{cases}$$

where $S = \sum_{i=1}^{D} X_i$. That is, Party A takes on the risk for a certain *tranche* of the losses, and will get paid for all income above the first α fraction until it reach the β fraction. (In finance parlance, looking at losses rather than income, $1 - \beta$ and $1 - \alpha$ are known as the "attachment" and "detachment" points of the tranche. If $\alpha = 0$ the tranche is called "senior", while if $\beta = 1$ the tranche is called "equity" sometimes also known as "toxic waste", tranches with $0 < \alpha < \beta < 1$ are known as "mezzanine".) We also consider the *binary CDO* variant F_{α}. that outputs 0 if $S < \alpha D$ and 1 if $S \geq \alpha D$. (These can correspond to discrete events such as credit rating change for the tranche.)

CDO's are an extremely popular financial derivative. One reason is that owing to the law of large numbers, even if the probability of default of each individual asset is not very small, if all of them are independent then the probability that a relatively senior tranche (with $1 - \beta$ larger than this probability) will lose money is extremely small. Hence these senior tranche and even some mezzanine tranches were considered a very safe investment and given high credit rating like AAA.

[8] For simplicity we consider a particular kind of a CDO - what's known a single tranche CDO. Also, at the current level of abstraction it does not matter what are the underlying assets, or if it's a so called "synthetic" or "asset-backed" CDO.

In particular CDO's are used to package together relatively high risk (e.g., subprime) mortgage loans and pass them on to risk-averse investors.

A.1 Our construction versus real-life derivatives

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

Role of random graph Real-life constructions do not use a random graph. Rather, seller tries to put together a diversified portfolio of assets which one can hope are "sufficiently independent." We are modeling this as a random graph. Many experts have worried about the "adverse selection" or "moral hazard" problem inherent because the seller could cherry-pick assets 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 F.4), which is perhaps more in line with economic practice in which even the senior most tranche has around 1% probability of incurring a loss (though of course for current parameters $\sqrt{\log D} \leq 3$).

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

B Omitted proofs from Section 2

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

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

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

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

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

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

B.1 Maximizing Profit in Binary Derivative Setting

We provide a more complete analysis than the one in Section 2 of the full optimization problem for the seller. Recall that his profit is given by

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

yet at the same time he is not allowed to put too many edges into the cluster (otherwise this dense subgraph can be detected). We abstract this as an optimization problem (let $\frac{b}{2\delta} = a$, $x_i = t_i/2\delta$):

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

As argued in Section 2, 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.

C Omitted proofs from Section 3

Theorem 3 (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 with an (m, n, d) subgraph is $\epsilon n + \tilde{\Theta}(\frac{\sigma}{D} mV)$, where $\epsilon = O(be^{-(b/2\sigma)^2/2}/\sigma)$.*

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.

□

D Omitted proofs from Section 4

Theorem 4 (Restated)**.** *In the binary CDO² setting, when $m^2 \gg \frac{M^2D^2}{N}$, $T = \frac{1}{2}(\mathrm{E}[poly] + \mathrm{E}[exp])$, the lemon cost for polynomial time buyer is $(1 - o(1))N/4$, while the lemon cost for exponential time buyer is only $N/4e^{-m^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant.*

Proof. From the results in Section 2 we know $\mathrm{E}[\mathrm{poly}] - \mathrm{E}[\mathrm{exp}] = \Theta(m)$. Therefore $\mathrm{E}[\mathrm{poly}] - T = \Theta(m)$ and $T - \mathrm{E}[\mathrm{exp}] = \Theta(m)$.

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, ..., x_N be the indicator variable for assets that x_i is 1 when i-th asset default. Then Y is a function of $\{x_i\}$, we denote this function $Y = f(\mathbf{x})$. Since each asset appears in $MD/N = N^{0.1}$ CDOs, $|f(\mathbf{x}) - f(\mathbf{x}')| \leq MD/N$ if $\mathbf{x}$ and $\mathbf{x}'$ differ by only 1 position. By McDiarmid's bound:

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

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 polynomial time buyers

$$\Pr[Y \geq T] = 1 - \Pr[Y < T] \geq 1 - \Pr[|Y - \mathrm{E}[\mathrm{poly}]| > m] \geq 1 - e^{-\Theta(m^2N/M^2D^2)},$$

similarly, for buyers with unbounded computation time,

$$\Pr[Y \geq T] \leq \Pr[|Y - \mathrm{E}[\mathrm{exp}]| > T - \mathrm{E}[\mathrm{exp}]] \leq e^{-\Theta(m^2N/M^2D^2)}.$$

The lemon cost for this CDO^2 is just the probability of loss multiplied by the payoff of CDO^2. Therefore, the lemon cost of exponential time buyer 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 the lemon cost of polynomial time buyer can be as large as $(1 - o(1))N/4$.

□

Theorem 5 (Restated)**.** *In the CDO^2 with tranching case, when $m^2d^2 \gg M^2D^2/N$, $T = \frac{1}{2}(\mathrm{E}[poly]+\mathrm{E}[exp])$, the lemon cost for polynomial time buyers is $\Theta(mdN/MD)$, while the lemon cost for exponential time buyer is at most $mdN/MDe^{-m^2d^2N/M^2D^2} = O(e^{-N^\epsilon})$ where ϵ is a positive constant.*

Proof. From Section 3 we know $\mathrm{E}[\mathrm{poly}]-\mathrm{E}[\mathrm{exp}] = \Theta(mdN/MD)$. Since $T = 1/2(\mathrm{E}[\mathrm{poly}]+\mathrm{E}[\mathrm{exp}])$, $\mathrm{E}[\mathrm{poly}] - T = T - \mathrm{E}[\mathrm{exp}] = \Theta(mdN/MD)$. 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, view Y as a function of $\mathbf{x}$, where x_i's are indicator variables for assets. 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 polynomial time buyers,

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

while for exponential time buyers

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

That is, polynomial time buyers almost always loss money, and their lemon cost is $(\mathrm{E}[\mathrm{poly}] - T) = \Theta(mdN/MD)$, 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 $e^{-\Theta(N^{0.5})}$). □

E Derivatives resistant to Dense Subgraphs

In this section we provide details for the dense subgraph resistant derivatives mentioned in Section 5.

E.1 XOR instead of threshold

As a warmup we give the more exotic of the two constructions, since its correctness is easier to show. Instead of simple sum of the variables, we use XOR or sum modulo 2. This is one of the famous functions used in computer science that resists local manipulations and is used in hashing, pseudorandomness, etc.

Theorem 6. *Suppose in our illustrative example we use XOR instead of simple sum, so that the product produces a payoff iff the XOR of the underlying variables is* 1*. When $D \gg \log M + \log 1/\epsilon$, a dense subgraph that can change the expectation of even a single particular output (derivative) by ϵ can be detected Ex-Post.*

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

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

Theorem 7. *When $D \gg \log 1/\epsilon$, dense subgraphs are Ex-Ante detectable if the ratio of manipulated output (m/M) is larger than the ratio of manipulated input n/N.*

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

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

so whether there is a dense subgraph can be detected by cycle counting algorithm (see Section F.2) □

E.2 Tree of Majorities

The XOR function is not too natural in real life. One particular disadvantage is that XOR function is not *monotone*. Buyers would probably insist upon the property that the derivative should be rise and fall with the number of assets that default, which is not true of XOR. Now we give a different function that is monotone but which still deters manipulation.

Let us think through the criteria that such a function must satisfy. Of course we would want the function to be more resistant to fixing a small portion of its inputs. And, the function needs to have symmetry, so that each input is treated the same as all other inputs. (We suspect that if the function is not symmetric then it makes manipulation easier.) A standard function that is monotone, symmetric and not sensitive to fixing a small portion of input is the tree of majorities:

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

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

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

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

Theorem 8. *Fixing any t inputs to 0, while other inputs are independent 0/1 variable with probability 1/2 of being 1, the probability that TM^k is 0 is at most $(1 + t/2^k)/2$.*

Proof. By induction.

When $k = 0$ the theorem is trivial.

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

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

□

Theorem 9. *Suppose in the illustrative example the financial products pay* 1 *iff the tree function* TM^k *(where* $D = 3^k$*) of the* D *underlying mini-assets evaluates to* 1*. If* $D \gg \log^3 M$ *then any manipulation that significantly changes the expectation of even a single financial product is ex-post detectable.*

Proof. By the above analysis, to control the output value of TM function, the dense subgraph parameter needs to satisfy $d = O(D^{2/3})$. Thus when $D \gg \log^3 M$ this is Ex-Post detectable because any of the financial products in which d of the inputs are 0 in the dense subgraph. □

F Detecting Dense Subgraphs

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. The goal is to quantify the range of dense subgraph parameters that will make the dense subgraph *are* detectable using efficient algorithms. The dense subgraph parameters used in our constructions fall outside this range.

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

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

F.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.

F.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.

F.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.

F.4 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. assets 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]

Ex-Post detection is easier than Ex-Ante detection, because we have more information after the result has been revealed: the junk assets must be assets in the set of the defaulted assets, and most of the manipulated derivatives are in the set of derivatives with no return. One simple use of this observation is: when the expected number of derivatives with no return in random graphs is

[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 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".

extremely small (e.g. $o(m)$), then almost all derivatives with no return are in the dense subgraph, and hence one can find it easily. But the number of derivatives with no return is large (e.g. $\Omega(M)$) then it still seems hard to find the dense subgraph even ex-post. We do not have a proof that this is the case, but can show a relation between finding the planted subgraph in the ex-post and ex-ante setting in a slightly differe nt model of random graphs.

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

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

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 $b = O(\sqrt{D})$.

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.

F.4.1 Ex post undetectability in a DeMarzo-type model

One can also modify our DeMarzo-type model so that seller's tampering is undetectable ex post.

Assume 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 asset class model, we need to be careful when picking the distribution of p_{class}. The easiest choice $p_{class} = 1/2$ for good classes and $p_{class} = 1$ for junk classes leads to ex post detection, because the assets in junk classes all defaulted, but for a good class, only with exponentially low probability all assets will default. Therefore, 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.

F.5 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

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 1. If it is not satisfied then a random graph is expected to have a graph as dense as the dense subgraph.

5. $\frac{MD^2}{N} \gg (\frac{md^2}{n})^2$.

   When this condition is satisfied, the dense subgraph is hard to detect for both cycle counting (Section F.2) and SDP (Section F.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 2

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 4 and Theorem 5