CS 620 Project: Computational Complexity and Financial Derivatives

Jeff Kinne

Indiana State University

jkinne@indstate.edu

November 18, 2010

Abstract

For this project I am studying a recent paper [ABBG10] that considers what effects computational complexity considerations have on the analysis of financial derivative products. The conclusion of [ABBG10] is that the products currently in use could potentially be manipulated by the sellers of the products to artificially decrease the amount they must pay out for the products when they mature. This result assumes the difficulty of a certain NP problem called the densest subgraph problem.

Both [ABBG10] and a later work [Zuc10] give some suggestions on designs for financial products that would circumvent the issues pointed out by [ABBG10].

Current Project Status

Certain parts of the paper have been difficult to process, so I am not as far along as I would have hoped. I have a high level understanding of each of the results in the paper, but I have not been able to verify all of the details yet. I just recently happened upon [Zuc10], so I have not yet carefully considered that paper.

Timeline on Remaining Work for Project

In the near term, I will continue to verify the details of [ABBG10], and will include some of the proofs in the final project writeup. [Zuc10] is also very interesting, so I will look at this paper as well. By the end of the project, I plan to have a good understanding of [Zuc10] so that I can include a description of the results from that paper in this project writeup as well. I plan to work on this project for an average of 8 hours per week for the remainder of the semester.

Self-Evaluation

I have spent a fair amount of time on this project, but I have encountered difficulties I did not foresee. Still, things would be better by now if I had started working in earnest earlier. I would give myself a B at this point (though I would likely give a student a higher mark for having completed the same work I have).

Suggestions

Whatever part of your project you currently understand, focus on taking those parts and writing them up. You can identify parts that you still need to figure out or look into further. By this point, you hopefully have the big picture and are working on understanding the details. So for the midpoint turnin, you can focus on writing up a nice introduction to your problem, stating whatever results you will understand the proof of, and give some of the details of some part of it if you understand them by now.

1 Introduction

Lemon Cost Suppose you are in the market for a car. You are a poor student, so you will be buying a used car. If the car works fine, it is worth \$5000 to you. But there is some probability that the previous owner did not take care of the car properly, and that the car breaks down soon after you purchase it. We call such a car a "lemon". If you know that 10% of all used cars are lemons, then the expected value of a used car that is chosen completely at random is $5000 \cdot .9 + 0 \cdot .1 = 4500$. This means that you might be willing to pay \$4500 rather than \$5000 for a used car. The difference between the expected value when no lemons are present and the expected value when lemons are present is called the *lemon cost*. In this example, the lemon cost is \$500.

Consider another example. If we are an investor in the housing market, we may wish to finance a mortgage of someone's house. But with some probability, the person will not be able to pay off their mortgage (due to losing their job, dying, etc.). To keep the calculations simple, suppose each homeowner is able to pay off their mortgage with probability 1/2 and defaults on the mortgage with probability 1/2. If they pay the mortgage, the investor is paid off in full; if the homeowner defaults, the investor gets nothing (actually, the investor could sell the house for a loss, but letting the investor get nothing makes the calculations simpler). If there are N houses under consideration, and each house will payoff 1 with probability 1/2 and 0 with probability 1/2, then the expected value of the collection of houses is $N/2$. Now suppose there are n lemon houses that will pay off 0 with probability 1. Then the expected value of the collection of houses is $1 \cdot (N-n)/2 + 0 \cdot n = (N-n)/2$, and the lemon cost is $n/2$.

1.1 Financial Derivatives

Rather than devaluing the collection of houses by $n/2$, can we do something so that the lemons have a smaller effect on the expected value? Financial derivatives are supposed to help reduce the lemon cost (financial derivatives have other purposes as well, as we will see). There are a number of different ways financial derivatives can be structured.

CDO A *collateralized dept obligation (CDO)* is defined as follows. There are D different assets $X_1, ..., X_D$ (you can think of these as houses if you like) that each pay off 1 with probability 1/2 and 0 with probability 1/2. Of course we could use different numbers than 1, 0, and 1/2, but we will stick with these, which are good enough to convey the ideas. There is some value T with $0 < T < D$ so that the investor buying the CDO will receive the first T payout from these assets and receives nothing above that. So if the assets combine to pay out at most T, then the investor

receives that amount; if the assets combine to pay out more than T, the investor still receives just T.

One reason the CDO is designed this way is to reduce risk. If an investor were to purchase one of the assets that pays off 0 with probability 1/2, that is very risky. On the other hand, if we set $T = D/4$ and let the investor invest in the CDO, then with very very high probability the investor will receive a full payout of T. This is because the sum of the D random variables will be very close to $D/2$ with very high probability (this can be proved using the Chernoff bound). The seller of the CDO gains from this arrangement as well because any income from the assets beyond T is kept by the seller, and also the seller has some "insurance" against large losses – if the number of assets paying off is extremely low, then the seller gets to keep the payment of the buyers.

Binary CDO A *binary CDO* is a simplification of a CDO so that it is easier to analyze. A binary CDO pays off some value V if at least T of the assets in the CDO are 1; the binary CDO pays off 0 otherwise. This makes the analysis easier because then the expected value of the binary CDO is just V times the probability the binary CDO pays off at least T. That is, the expected value is $V \cdot \Pr[\sum_{i=1}^{D} X_i \geq T]$. Using the Chernoff bound, if we were to set $T = \frac{D}{2} - c\sqrt{D \log D}$ for a large enough constant c, we will make sure that the probability is at least $1 - \frac{1}{D^{c'}}$ for another constant c', and then the expected value of the binary CDO is close to V.

What would the lemon cost be of introducing d lemons into the binary CDO? This would mean that d of the assets always evaluate to 0 and $D - d$ of the assets evaluate to 0 or 1 with equal probability. What is the probability that at least T of these $D - d$ assets evaluate to 1. If we keep $T = \frac{D}{2} - c\sqrt{D \log D}$ and set $d = 2c\sqrt{D \log D}$, then the expected number of assets that evaluate to 1 (which is $\frac{D-d}{2}$) will be equal to T; further, the probability the number of assets that evaluate to 1 is less than T will be roughly 1/2, and the expected value of the binary CDO will be roughly $V/2$ – which is much lower than in the case with no lemons.

On the other hand, if we had set $d = c\sqrt{D \log D}$ with the same value of T, then the expected number of assets that evaluate to 1 would be $T + c\sqrt{D \log D}/2$; further the probability of being less than T would still be $\frac{1}{D^{O(1)}}$. Thus the parameters can be set to minimize the lemon cost for binary CDO's.

Portfolio of CDO's In practice, a common financial derivative is a portfolio of CDO's. There are N assets on which to base the portfolio of CDO's, and the seller wishes to create M different CDO's out of these assets. Each CDO is composed of D different assets. The parameters are often set with $M \cdot D > N$ so that each asset appears in more than one of the M CDO's. The seller would then choose at random how to distribute the assets to the different CDO's. This type of financial derivative and other related types are the main focus of [ABBG10].

2 Main Results of [ABBG10]

The portfolio of CDO model of financial derivative has the feature that the seller needs to decide which assets to place in which CDO's. The main result of [ABBG10] is that in the presence of lemons the seller can choose the placement to decrease the expected payouts of the CDO's without computationally bounded buyers being able to detect this. The assumption is that the seller knows

exactly which assets are lemons and which are not; the buyer does not. Furthermore, the buyer is assumed to have access only to polynomial-time algorithms in attempting to detect tampering. Tampering is defined as allocating the assets to the individual CDO's in any way other than selecting them at random. For example, the seller could try to put the lemons into just a few of the CDO's to try and make sure those CDO's do not pay out.

[ABBG10] show that assuming a certain NP problem is hard to compute, any computationally bounded buyer will not be able to determine whether the seller has tampered with the makeup of the portfolio of CDO's or not. Without being sure which is the case, the buyer must assume the CDO's are tampered with and then therefore would only be willing to pay a lower price for the individual CDO's – so the lemon cost is increased in this case . An unbounded buyer on the other hand would be able to detect the tampering and could therefore verify that the buyer has not tampered with the makeup of the CDO's, and the lemon cost would be lower.

The main point of the paper is that the financial product behaves differently depending on whether we assume the buyer is computationally bounded or not, and that the reason this is the case is because the seller has information that the buyer does not. They also discuss how the payout rules could be modified to remedy this situation; essentially, rather than paying out if at least a certain number of assets evaluate to 1, instead a much more complicated formula would be used.

The precise statements within [ABBG10] are very parametrized, and these will be stated and discussed in the final version of this project.

3 Main Results of [Zuc10]

In a followup paper, [Zuc10] gives an approach to creating "pseudorandom" financial derivatives that would be immune to the type of tampering considered in [ABBG10]. I will give the precise statement of the results and the proofs in the final version of this paper.

References

[ABBG10] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products (working paper). In *Innovations in Computer Science*, pages 49–65, 2010.

[Zuc10] David Zuckerman. Pseudorandom financial derivatives, 2010. Working paper, 11 pages.