Algorithmic Tacit Collusion Is A Limited Threat To Competition

By Ai Deng (December 10, 2019, 4:55 PM EST)

Ever since the publication of Professors Maurice Stucke and Ariel Ezrachi’s "Virtual Competition," collusion by autonomous robots has been a hot topic in the global antitrust community. Indeed, last year, the Federal Trade Commission devoted an entire public hearing to the topic of artificial intelligence and its implications on competition.

A recent study by four economists has undoubtedly refueled both public and academic interest in algorithmic collusion.[1] In this study, the researchers found that their “algorithms consistently learn to charge supra-competitive prices, without communicating with one another ... this finding is robust to asymmetries in cost or demand, changes in the number of players, and various forms of uncertainty.”

Not surprisingly, this research has since been reported in a number of outlets including the Wall Street Journal, the Financial Times and Massachusetts Institute of Technology's Technology Review. What does the study say about the likelihood of algorithmic collusion in the real world? Does it fundamentally change how we should think about tacit collusion in general? These are the questions I want to address in this article.

The AI Pricing Algorithm

The researchers of the study used a reinforcement learning pricing algorithm.[2] The hallmark of an RL algorithm is learning by trial and error. Because the underlying premise of RL is to start with (nearly) no knowledge about the task at hand or the environment in which it operates, an RL algorithm needs to explore in order to figure out how to behave appropriately within that unknown environment.

As the algorithm explores, it receives rewards and punishments. It is through this type of feedback that it learns.[3] For strategic tasks, such as pricing in a competitive marketplace, the rewards a pricing algorithm receives depend on not only its own actions but also the competitors’.

As with any AI or machine-learning algorithm, it needs to be “trained.” A common way to train an RL algorithm in a strategic setting is known as self-play, which intuitively means that the algorithm is trained to learn to play against itself (i.e., the same algorithm). Self-play is how the researchers of the study trained their RL pricing algorithm.

Economic Environment

To understand how their RL algorithm sets prices, the researchers created an artificial marketplace in their computer lab. In this artificial marketplace, they specified the number of competitors and products as well as the demand for and the supply of these products. As a result, the competitive and the monopoly prices could be calculated with mathematical precision. This knowledge allows the researchers to tell if their RL algorithm learned to price competitively or collusively.

Because tacit collusion is possible when competitors interact with each other over time, the
researchers let the competing pricing algorithms do so. Such an experimental environment allowed the researchers to observe how their algorithm reacted to a competitor’s deviation from supra-competitive prices.

One of the most interesting observations in this study is that their RL algorithm appears to have learned to punish the “cheater” and reward the “collaborator.”[4] This type of reward-punishment strategy has been labeled as problematic collusive behavior.[5] And it is this observation that led the researchers to conclude that the RL algorithm has learned to tacitly collude.

Lessons from the Research

The finding of the RL algorithm’s robust tendency to tacitly collude is concerning. A common caveat for such an experimental study, however, is that the artificial market is too simplistic relative to any real market. That caveat aside, this research study crystallizes a number of other points that I have raised previously as to why the advent of algorithm tacit collusion in the real world is not as imminent as one might think, and perhaps more importantly, that we are not hopeless in combating algorithmic collusion.

Let’s start by noting that the RL algorithm in the researchers’ experiments takes an average of 850,000 periods of training to learn to “tacitly collude.”[6] Although that amounts to less than one minute of CPU time, it means that, in the real world, the algorithm “learns” after over 2,300 years if they change prices daily, or 97 years if they change prices every hour. And this is after the researchers limited the set of possible prices the algorithm could choose from.[7]

It is unlikely that any company would be amenable to allowing the algorithm to engage in trial-and-error with actual prices, let alone for decades. I will return to this point later. Recognizing this limitation, the researchers focused entirely on the algorithmic behavior after the training process had completed.[8] In the researchers’ words, this means that the learning happens “off the job.” But if the algorithm is indeed trained off-the-job, several practical considerations that I raised previously become immediately relevant.[9]

When the training is done offline in a lab environment, chances are that the developers already know the algorithmic behavior before it is deployed. As is the case with any type of software development, extensive testing would be done and typically documented. This means that, to the extent an algorithm learns to tacitly collude in off-the-job training, there would typically be a paper trail that could later be discovered in agency investigation and in private litigation.

Indeed, the researchers of the study designed an experiment to show that the algorithms have learned to tacitly collude after training. This is arguably one of the most significant contributions of their study. Equipped with such an experimental approach, one can then uncover algorithmic tacit collusion in the experimental stage. Furthermore, in the event that the algorithm developers intend to develop a tacitly collusive algorithm, there may even be hints of that intent in their promotional and marketing materials. In the presence of such documents, firms would have a hard time placing the blame on a collusive algorithm.

If placing blame on a robot is unlikely to be a winning argument when the training is done off the job, what if we move the learning on the job?[10] Let’s further assume that technological advances make the learning much more efficient, so the training time is less of a factor.

In this case, one particularly relevant reality is that there is no guarantee that companies adopting an algorithm would necessarily face competitors using another algorithm, let alone the same algorithm. In fact, the competitors may not use algorithms at all.[11]

It is well known in the AI field that uncertainty about your opponents presents a challenge in eliciting cooperative behavior. Intuitively, that is because a good algorithm must now be made flexible: It needs to learn to cooperate with others without necessarily having prior knowledge of their behaviors. But to do that, the algorithm must be able to deter potentially exploitative behavior from others and, “when beneficial, determine how to elicit cooperation from a (potentially distrustful) opponent who might be disinclined to cooperate.”

The researchers of this study recognized this difficulty when they acknowledged that “[i]n reality,
there are many different forms of reinforcement learning, and Q-learning algorithms [the RL algorithm used by the researchers] themselves come in different varieties. It would therefore seem necessary to extend the analysis to the case of heterogenous algorithms more systematically. “[12]

**Future Extensions**

This important research can be extended in many directions. In this section, I highlight three extensions that will further our understanding of algorithmic collusion.

First, future research on algorithmic collusion should tackle the speed of the (on-the-job) learning head on. As I discussed above and elsewhere, a collusive algorithm is arguably irrelevant to the antitrust community if it takes an unrealistically long time to learn to collude.

In fact, the longer the learning takes, the more likely the market structure will change during the learning stage. For example, the number of competitors may change due to entries and exits; new technologies may emerge and disrupt an industry; even the macroeconomic environment may change, all of which create a nonstationarity environment, making it difficult for an algorithm to learn.

Second, future research should explore how the pricing algorithm behaves in an environment where the demand is noisy and the competitors cannot observe each other’s prices perfectly. While this study looked at noisy demand, it assumes that the algorithms know and remember each other’s past prices perfectly. Imperfect monitoring of competitor prices is not only common in most marketplaces but also has been shown to disrupt collusive pricing.

Third, future research should further explore how the memory (i.e., how long the algorithm remembers the pricing history) impacts the algorithm’s behavior. The study shows that their algorithms lose 33% of the profit if they remember the prices in the past two periods as opposed to one. In other words, it seems that the algorithms would need to be forced to erase some memories if they want to achieve a higher profit closer to the monopolistic level.

**Conclusion**

Can an algorithm learn to tacitly collude? The answer is a resounding yes in experimental environments. [13] The recent economic study that I discussed in this article is another important piece of evidence. What sets this study apart from others is the demonstration that simple reinforcement-learning algorithms, whose goal is to simply maximize long-term payoff, learn to tacitly collude with one another. They do so without communication and without being explicitly programmed or designed to do so.

At the same time, this research also highlights some practical obstacles to algorithmic collusion and why we are not as hopeless as one might think. I am also encouraged by the opportunities for algorithmic compliance and in particular, the role of explainable AI in algorithmic compliance. [14]

---

Ai Deng, Ph.D., is an associate director at NERA Economic Consulting, an editor of the American Bar Association’s Antitrust Source, and a lecturer of economics at Johns Hopkins University’s Advanced Academic Program.

The opinions expressed are those of the author(s) and do not necessarily reflect the views of the firm, its clients, or Portfolio Media Inc., or any of its or their respective affiliates. This article is for general information purposes and is not intended to be and should not be taken as legal advice.


[3] The most well-known RL algorithm is probably DeepMind’s AlphaGo Zero that has beat the world
champions in the ancient game of Go. It gets the moniker Zero because the algorithm started with zero knowledge about the game of Go.

[4] To show this, the researchers forced their algorithm to “cheat” in one period by lowering its own price. They found that the competitor would immediately lower its price in the next period by even a larger amount, suggesting a “punishment.” What is also interesting is that it appears that the “cheating” algorithm anticipates the punishment because it also immediately lowers its price. After that, the researchers find that both algorithms gradually raise their price to a pre-cheating supra-competitive level in almost lock-steps.

[5] See, for example, Joseph E. Harrington, Developing Competition Law for Collusion by Autonomous Artificial Agents, J. of Com Law & Econ. 14(4).

[6] The learning can take as low as 400,000 and as long as up to 2.5 million periods, supra note 2, at 22.

[7] In their baseline experiment, the researchers allow only 15 different choices, supra note 2, at 21.

[8] “The downside of such relatively simple AI technology is the slowness of learning. But this is not a problem as long as one focuses on the behavior of the algorithm once they have competed the training process. This paper takes precisely such off-the-job approach. That is, we conform to the standard practice of training the algorithm before assessing their performance.”


[10] In fact, the researchers have acknowledged several benefits of on-line training (“On-the-job learning seems necessary, in particular, when the economic environment is changeable, or when the training environment does not exactly reflect the reality of the markets where the algorithms are eventually deployed.”)

[11] I assume that firms do not collude to adopt the same algorithm.

[12] Supra note 2, at 39. The researchers also recognized that “whether Q-learning algorithms [the RL algorithm used by the researchers] could be exploited during the learning phase, if it were conducted on-the-job, is an interesting and difficult question for future study,” supra note 2, fn 32.
