Impact of Internet on Business Activities in Serbia and Worldwide

Uticaj Interneta na poslovanje u Srbiji i svetu

DOI: 10.15308/SINTEZA-2014-66-70

MAXIMAL PAYOFF STRATEGY FOR VICKERY AUCTION USING GAME THEORY AND COMPUTER ALGEBRA SYSTEMS

Miroslav D. Lutovac[1], Aleksandra M. Lutovac[2]
[1]Singidunum University,
[2]EKOF University of Belgrade

Abstract:
The theory of auction was first formalized by William Vickery. His classical contribution was one of the standards in the Game theory and it is contained in standard textbooks. This is an auction in which the highest bidder wins but pays only the second-highest bid. This variation over the normal bidding procedure is supposed to encourage bidders to bid the largest amount they are willing to pay. Optimization of payoff strategy for Vickery auction can be programed using game theory and computer algebra systems, as it is presented in this paper.

Key words:
auction,
strategy;
game theory,
programming;

This work was supported by the Ministry of Education and Science of Serbia under Grant TR 32023.

INTRODUCTION

In 1994 the United States government adopted an auction in selling the right on the radio frequency for pager service providers. Since then the governments all over the world have adopted auctions in selling the newly invented rights for the private corporations. The theory of auction was first formalized by William Vickery in 1961. In his original work on auction theory, William Vickrey pointed out several types of auctions used in practice [1]. His classical contribution is one of the standards in the Game theory. Applying game theory to the study of auctions, Vickrey showed that some auctions are strategically equivalent and how to compute Nash equilibrium bid strategy for bidders [2].

The terms Vickrey auction and the second-price sealed-bid auction are equivalent and used interchangeably. The Vickrey auction has been widely advocated for multi-agent systems [3]. eBay's system of proxy bidding is similar to Vickrey auction. A variant of a Vickrey auction, named generalized second-price auction, is used in Google's and Yahoo!'s online advertisement programs [4], [5]. There were attempts to construct the payoff matrix for two and more player game by simulation approach, and to show that the weak dominance holds for the second price auction [6].

Papers written on auctions are usually described using mathematical notation [7], and are not applicable for practical usage. The main purpose of this paper is to show that programming of auctions is not difficult using Computer Algebra Systems (CAS) [8]. Even more, inexperienced user can use preprogramed code for simple analysis and optimization.

COMPUTER ALGEBRA SYSTEMS

Computer Algebra System Features

A computer algebra system is a software program that allows computing with mathematical expressions in a way which is similar to the traditional handwritten computations of the mathematicians and other scientists. Wolfram Research brought into the mainstream the original software Mathematica in 1988 as the first computer algebra system [8]. The main computer algebra system features are: symbolic handling of arbitrary formulas, exact and arbitrary-precision arithmetic, symbolic expansion, factoring, simplification and substitution, symbolic integration, differentiation, summation, limits and series, symbolic solvers for systems of equations, differential equations, and difference equations, full support of elementary and special functions, exact and symbolic matrix operations.

The software is integrated with MathWorld's encyclopedic website of mathematical information. Thousands of ready-built interactive demonstrations of mathematical concepts can be used as starting point for specific applications. Web interface is provided using webMathematica. Some other software companies offer valuable CAS solutions. Maxima is a descendant of Macsyma, the legendary computer algebra system developed in the late 1960s at the Massachusetts Institute of Technology. Texas Instruments came with a computer algebra system based on Derive, and was one of the first calculators to offer 3D graphing. MuPAD is a computer algebra system that was purchased by MathWorks and the MuPAD code was included in the Symbolic Math Toolbox add-on for MATLAB. Maple is a commercial computer algebra system developed and sold commercially by Maplesoft.

Financial Engineering with Mathematica

An extraordinary encyclopedia of option pricing techniques is presented in [9]. Authors from the Hebrew University have published several articles in Mathematica in Education and Research. Much of this material had its origins in courses on financial engineering [10].

In some papers it is described a step-by-step procedure for developing code for some economic functions. It starts with the simplest case and the user can add new more complicated features, such as more players in the auction game simulation [6]. This can be confusing for inexperience users. Actually, some very complex procedures can be programmed with just a few code lines using powerful built-in CAS functions, as it is presented in this paper.

Mathematica has fully integrated support for many of the tools used in classical and modern finance. These capabilities include financial instrument valuation (FinancialDerivative, FinancialBond), advanced time value of money computations (TimeValue, Cashflow, EffectiveInterest, AnnuityDue, Annuity), and advanced financial charting with a library of technical indicators (InteractiveTradingChart, TradingChart, CandlestickChart). Mathematica also provides immediate access to a large array of financial and economic data, and contains financial import and export tools for working with external data. More information on the Vickery auction and game theory is provided by Wolfram Alpha.

Knowledge based system and applications using CAS

Many researchers start their work by studying theory in order to get better insight into observed phenomena. Sometimes they cannot get the numeric values of parameters presented in the published reports. This is even more complicated when the theory is statistical and there are no closed form deterministic solutions. An original approach and method to analyzing mathematically written papers, such as Expectation-Maximization (EM) algorithm is presented in [11].

An original algorithm that combines design and synthesis step, using computer algebra system, with the optimization for targeted implementation technology is presented in [12]. The approximation step is based on Gegenbauer polynomial and the corresponding cost function. The knowledge, in the form of mathematical expressions, is typed into CAS and thereafter is automatically performed optimization using specified criteria.

The straightforward procedure for the design and analysis of uniform and nonuniform digital systems based on approximately linear phase IIR filters and frequency response masking technique (FRM) are derived using CAS [13].

The same approach can be used for developing any algorithm, such as the aeronautical communication system that have to provide more communications capacity and to increase capabilities [14]. The improvements are necessary to be able to cope with the expected air traffic growth in future. The main idea is to automate the design procedure starting from the block diagram of the system and carrying out the implementation code on the target hardware. The role and importance of symbolic computation in communication systems is exemplified on OFDM (Orthogonal Frequency Division Multiplexing). The development tools are Mathematica, and application software SchematicSolver [15].

The market for telecommunications networks and services is constantly changing. So is the regulatory approach. Technological progress creates new challenges for the creation of a level playing field between traditional and new operators and service providers. CAS can be used to analyze and develop new applications and services that can be more robust to economic impacts [16].

ALGORITHM OF VICKERY AUCTION

In this paper we will use algorithm described in [6]. Vickery auction is the second price auction, in which the winner is the highest bidder, while the winner pays the second highest bid price to acquire the object. The true valuations of the object are denoted by v_i, and this value can be different for different i bidders. The strategy sets are $s_i \in S_i = \{10, 20, ..., 10n_i\}$, where $10<v_i<10\ n_i$. When player $A1$ selects a strategy, s_1 and player $A2$ selects a strategy s_2 from the set S_i, and $s_1>s_2$, player $A1$ is the winner, and $A1$ pays s_2 to acquire the object. In that case, the payoff of player $A1$ is $p_1=v_1-s_2$ and the second payoff is $p_2=0$.

Suppose that $v_1=40$ and $v_2=20$, and the bidding is selected from the strategy sets $s_1=60$, and $s_2=50$. The winner is $A1$ because $s_1>s_2$, and the payoffs are $p_1=40-50=-10$, and $p_2=0$. Obviously, player $A1$ pays 10 more than the true value for player $A1$.

In another case, $v_1=40$ and $v_2=20$, and the bidding is selected from the set $s_1=60$, and $s_2=30$. The winner is $A1$ because $s_1>s_2$, and the payoffs are $p_1=40-30=10$, and $p_2=0$. The player $A1$ pays 10 less than his true value.

Suppose that $v_1=40$ and $v_2=20$, and the bidding is selected from the strategy sets $s_1=20$, and $s_2=30$. The winner is $A2$ because $s_2>s_1$, and the payoffs are $p_1=0$ and $p_2=20-20=0$. The player $A1$ loses this time.

The strategy of bidding with lower value than the true value is better because the payoff is larger, but the probability to win is much lower. The strategy of bidding with highest value than the true value is better because it is more promising to win, but the payoff can be lower and even negative – paying more than the true value.

The special case is when two or more bidders have the same strategy value that is the largest value, for example $s_1=s_2$. In this case, the winner is selected by lottery, so that the probability for each player becomes the winner is 1/2. If the winner is player *A*1, *A*1 pays $s_1=s_2$ to acquire the object, and the payoff of player *A*1 is the expected value, $p_1=(v_1-s_1)/2$, while the second player *A*2 has the expected value $p_2=(v_2- s_2)/2$.

Suppose that $v_1=40$ and $v_2=20$, and the bidding is selected from the strategy sets $s_1=30$, and $s_2=30$. The winner can be *A*1 or *A*2 because $s_1=s_2$, and the payoffs are $p_1=(40-30)/2=5$, and $p_2=(20-30)/2=-5$. Obviously, payoff of player *A*1 is positive, while payoff of the second player is negative; *A*2 pays more than his true value.

In this paper we are describing how to program and play this auction in an attempt to determine the optimal bidding strategy with the highest payoff.

PROGRAMMING WITH MATHEMATICA

The code for calculating the payoff is defined as a function, **payoffVickery**, that has two arguments, the true valuations of the object **vABCD**, and the bidding prices **sABCD**, where arguments are matrices. The length of matrices is equal to the number of bidders. This means that the presented code works for arbitrary number of players, where the minimal number of players is 2.

```
payoffVickery[vABCD_,sABCD_]:=Module[
 {i,oABCD,nABCD,lABCD,pABCD=0 vABCD},
  oABCD=Ordering[sABCD];
  nABCD=Last[Tally[sABCD[[oABCD]]]][[2]];
  lABCD={oABCD[[-1]],oABCD[[-2]]};
  Do[pABCD[[oABCD[[-i]]]]=(vABCD[[oABCD[[-
i]]]] -
   sABCD[[lABCD[[2]]]])/nABCD, {i,nABCD}];
  pABCD]
```

The command **Ordering** sorts all values. The number of equal largest bidds is stored in **nABCD**. The two largest bidds are computed as **lABCD**. The payoff is calculated using **Do** loop, only for largest bids. The auctioneer wants to sell and get the highest possible payment; this can be computed in a similar manner using another function, **eVickery**.

```
eVickery[vABCD_,sABCD_]:=Module[
{i,oABCD,nABCD,lABCD,eABCD=0 vABCD},
  oABCD=Ordering[sABCD];
  nABCD=Last[Tally[sABCD[[oABCD]]]][[2]];
  lABCD={oABCD[[-1]],oABCD[[-2]]};
  eABCD=sABCD[[lABCD[[2]]]]]
```

Another important function is defined for computing random value for each player from the set of preferred strategies, **sRandomInteger**.

```
sRandomInteger[x_,xABCD_]:=
   10 RandomInteger[{1,xABCD+x/10}]
```

At the beginning of the program, we can define the true values, **vABCD**, number of games, **nsteps**, and the initial matrices for storing results of each single game.

```
vABCD={60,30,50,20,40};
nsteps = 4 10⁴;
smX={}; pmX={}; emX={};
```

This is the only knowledge that describes the Vickery auction. Once stored in the CAS, the knowledge can be used for computing any single auction, or a number of auctions with different strategies based on arbitrary probabilities. In order to prevent errors of inexperience user, the help and test instructions can be added in a similar way as in [15].

It is important to notice that the simplicity of the code is due to the powerful built-in functions of CAS, and the newly defined functions for auction can be used as they are built-in. The new knowledge can be put at the beginning of working notebook, or it can be stored as an additional knowledge in a package that can be inputted into CAS by a simple input command.

USING THE KNOWLEDGE FOR OPTIMIZATION

The code for different number of cases can be obtained by iterative evaluation of the defined functions for all possible values of strategies, say for i from the range -2 to 5.

```
xRange=Table[i,{i,-2,5}];
Do[{
 xABCD = {xRange[[j]], 1, 2, 0, 1};
 SeedRandom[1];
 sABCD =
   Map[sRandomInteger[vABCD[[#]], xAB-
CD[[#]]] &,
   Table[i, {i, Length[vABCD]}]];
 pABCD = payoffVickery[vABCD, sABCD];
 eABCD = eVickery[vABCD, sABCD],
   …},
{j, Length[xRange]}]
```

In this case, all possible strategies are i ∈ {-2, -1, 0, 1, 2, 3, 4, 5}, where the corresponding bidding prices are 10i. It is interesting to notice that the random value is generated using the command `Map`.

```
Map[sRandomInteger[vABCD[[#]],xABCD[[#]]] &,
```

In order to make sure we get the same sequence of pseudorandom numbers on different occasions, the command `SeedRandom[n]` resets the pseudorandom generator, using `n` as a seed.

As an example, we can evaluate 40 000 iterations for randomly generated bidding prices with uniform distribution, with expected true values v_1=60, v_2=30, v_3=50, v_4=20, and v_5=40, that is defined as **vABCD={60,30,50,20,40}**.

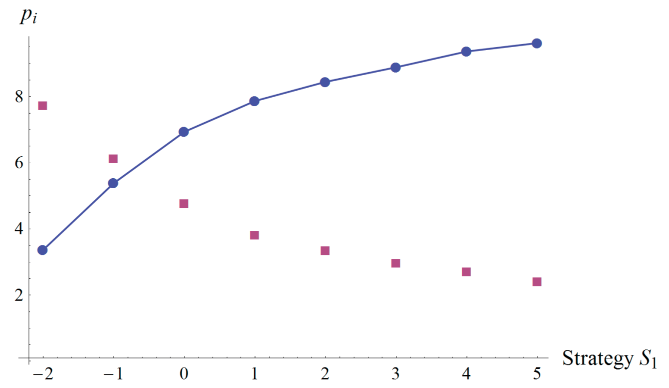

Fig. 1. Payoff p_1 (solid line-circuils) and p_3 (squares) for different strategies S_1.

The different strategies are defined only for the first bidder $i \in \{-2, -1, 0, 1, 2, 3, 4, 5\}$, that is $s_{1\max} = v_1+10i$, or from the set of maximal bidding prices $s_{1\max} \in \{40, 50, 60, 70, 80, 90, 100, 110\}$. For all other bidders, the strategy is to have only one value, $s_{2\max}$=30+10=40, $s_{3\max}$=50+20=70, $s_{4\max}$=20, $s_{5\max}$=30.

The actual bidding price is randomly selected from the range $10 \le s_i \le s_{i\max}$, with the discrete uniform distribution. Fig. 1 shows that payoff of the bidder with the highest true value v_1 increases with higher $s_{1\max}$. The payoff of the bidder with the second highest true value v_3 decreases with higher $s_{1\max}$.

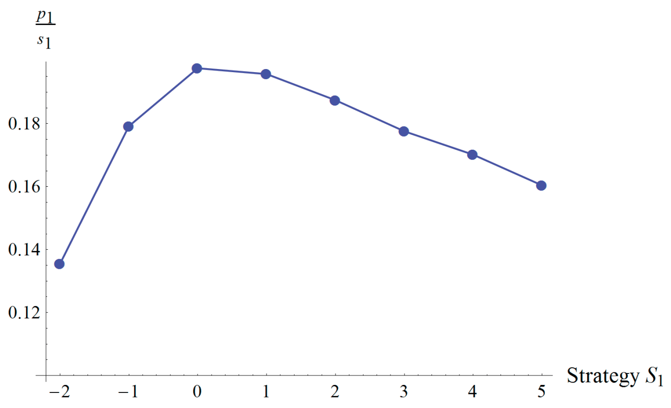

Fig. 2. Ratio of $p_{1/}\ s_1$ for different strategies S_1.

For the bidder with the highest true value v_1, the ratio of the payoff and the true value has a maximum for the strategy 0, that is $s_{1\max} = v_1$, as it is presented in Fig. 2. This can lead to a conclusion that the highest bid should not be larger than the true value that is an auction without a risk.

For the bidder with the second highest true value v_3, the ratio of the payoff and the true value is a decreasing function for higher $s_{1\max}$, as it is presented in Fig. 3.

Let us define the difference of payoffs between two consecutive strategies,

$$\Delta p(\text{Strategy } S_i) = p(\text{Strategy } S_i) - p(\text{Strategy } S_{i-1}) \quad (1)$$

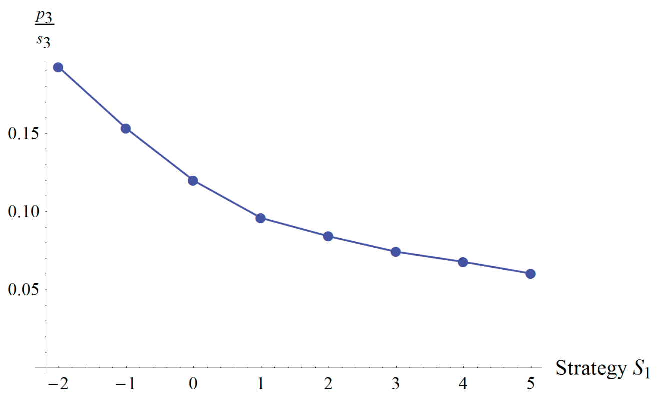

Fig. 3. Ratio of $p_{3/}\ s_3$ for different strategies S_1.

Fig. 4 illustrates that strategy 2 is probably the optimal because the strategy 3 and 4 are more risky and without considerably increased payoffs.

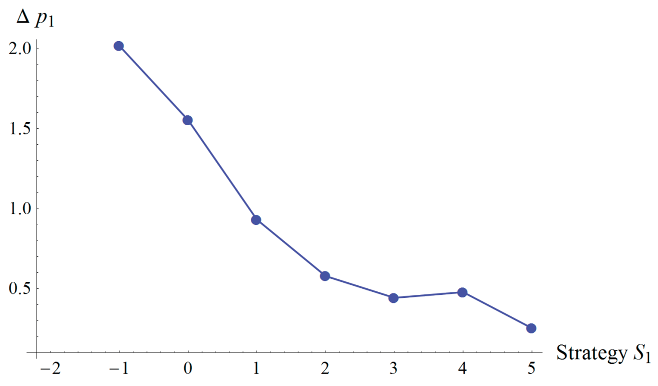

Fig. 4. Δp_1(Strategy S_1) for different strategies S_1.

The payment e of the auctioneer is also increasing function when bidders take a highest risk, and the expected payment is presented in Fig. 5.

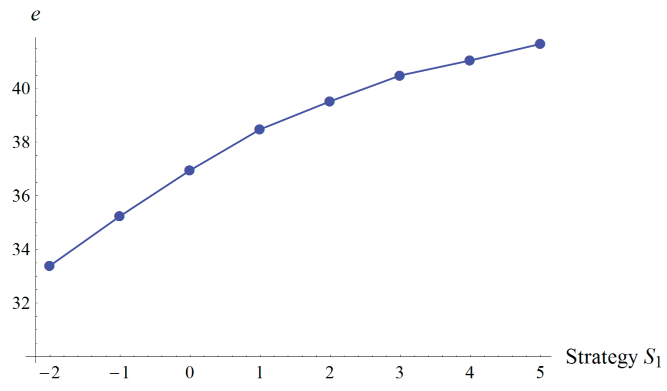

Fig. 5. Payment for different strategies S_1.

After determining the strategy for the bidder with the highest true value, the procedure can be repeated for all other players. Even more, the same program can be used for different expected true values of each bidder, and the best strategy can be selected as the most preferable solution. It should be noticed that the bidding price is randomly generated and that strategy determines the highest value of the strategy set.

CONCLUSION

The Vickery auction for arbitrary number of players can be easily programed using powerful commands of computer algebra systems. Assuming different strategies of players with the expected true values, the optimal strategy can be easily computed for different bidding prices.

REFERENCES

[1] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," The Journal of Finance, vol. 16, no. 1, pp. 8-37, March, 1961.

[2] D. Lucking-Reiley, "Vickrey Auctions in Practice: From Nineteenth-Century Philately to Twenty-first-Century E-commerce," Journal of Economic Perspectives, vol. 14, no. 3, pp. 183–192, Summer 2000.

[3] T. W Sandholm, "Issues in computational Vickrey auctions," Interna-tional Journal of Electronic Commerce - Special issue: Intelligent agents for electronic commerce, vol. 4, no. 3, pp. 107-129, March 2000.

[4] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords," American Economic Review, vol. 97, no. 1, pp 242–259, 2007.

[5] H. R. Varian, "Position Auctions," International Journal of Industrial Organization, 2007, vol. 25, pp. 1163–1178, December 2007.

[6] T. Fukiharu, Tools for Economic Research and Education, Faculty of Economics, Hiroshima University, English translation from manuscripts written in Japanese, http://home.hiroshima-u.ac.jp/fukito/Englishindex.htm

[7] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, Algorithmic Game Theory, Cambridge University Press, December 2007, pp. 559-564.

[8] S. Wolfram, The Mathematica Book, Cambridge: Cambridge University Press, 2003.

[9] W. T. Shaw, Modelling Financial Derivatives with Mathematica , Cambridge: Cambridge University Press, 1999.

[10] S. Benninga and Z. Wiener, "The Binomial Option Pricing Mode," Mathematica in Education and Research, vol. 6, no. 3, 1997.

[11] V. Mladenović, M. Lutovac, D. Porrat, "Symbolic Analysis as Universal Tool for Deriving Properties of Non-linear Algorithms – Case study of EM Algorithm," Acta Polytechnica Hungarica, vol. 11, no. 2, pp. 117-136, February 2014.

[12] M Lutovac, V Pavlovic, M Lutovac, "Automated Knowledge Based Filter Synthesis Using Gegenbauer Approximation and Optimization of Pole-Q Factors," Electronics & Electrical Engineering, vol. 19, no. 9, pp. 97-100, 2013.

[13] J. D. Ćertić, M. D. Lutovac, and L. D. Milić, "Approximately Liner Phase, IIR Digital Filter Banks," Telfor Journal, vol. 5, no. 2, pp/ 107-112, 2013.

[14] M. Lutovac, V. Mladenović, and M. Lutovac, "Development of Aeronautical Communication System for Air Traffic Control Using OFDM and Computer Algebra Systems," Studies in Informatics and Control, vol. 22, no. 2, pp. 205-212, June 2013.

[15] M. Lutovac, D. Tošić, SchematicSolver Version 2.2, 2010. Available: books.google.com/books?id=9ue-uVG__JsC

[16] V. D. Pavlovic, V. Mladenovic, M. Lutovac, Computer Algebra and Symbolic Processing in Modern Telecommunication Applications: A New Kind of Survey, in J. P. Barringer, ed, Telecommunications: Applications, Modern Technologies and Economic Impact, Nova Science Publishers, 2014.