



# A program to find all pure Nash equilibria in games with n-players and m-strategies: the Nash Equilibria Finder – NEFinder

*Um programa para encontrar todos os equilíbrios de Nash puros em jogos com n-jogadores e m-estratégias: o Nash Equilíbrio Finder – NEFinder*

Renan Henrique Caviccholi Sugiyama<sup>1</sup> , Alexandre Bevilacqua Leoneti<sup>1</sup> 

<sup>1</sup>University of São Paulo, School of Economics, Business Administration and Accounting, Research Group in Decision Sciences, Ribeirão Preto, SP, Brasil. E-mail: ableoneti@usp.br

**How to cite:** Sugiyama, R. H. C., & Leoneti, A. B. (2021). A program to find all pure Nash equilibria in games with n-players and m-strategies: the Nash Equilibria Finder – NEFinder. *Gestão & Produção*, 28(3), e5640. <https://doi.org/10.1590/1806-9649-2021v28e5640>

**Abstract:** Nash equilibrium is an important concept for studying human behavior in group decision making process. Given the complexity of finding Nash equilibria, computational tools are necessary to find them. Several programs were developed for this task. However, available programs are either not comprehensive or might be of difficult installation and handling, creating a “barrier of entry” to non-specialists. The aims of this research are twofold: (i) firstly, it was to identify and to discuss about the available programs for finding Nash equilibria; and (ii) secondly, based on the theoretical proprieties of a Nash equilibrium, to develop a program capable of finding all pure Nash equilibria in games with “n” players and “m” strategies (“n” and “m” being finite numbers) as a Macro tool for Microsoft Excel®. It is expected that the program can contribute to the area of Operations Research by providing a new tool that facilitates the use of game theory concepts within group decision-making problem-solving scenarios enabling practical applications using a widespread software.

**Keywords:** Group decision-making; Game theory; Nash equilibrium; Software.

**Resumo:** O equilíbrio de Nash é um conceito importante para estudar o comportamento humano no processo de tomada de decisão em grupo. Dada a complexidade de se encontrar equilíbrios de Nash, ferramentas computacionais são necessárias para encontrá-los. Vários programas foram desenvolvidos para essa tarefa. No entanto, os programas disponíveis não são abrangentes ou podem ser de instalação e manuseio difíceis, criando uma “barreira de entrada” para não especialistas. Os objetivos desta pesquisa são dois: (i) primeiramente, foi identificar e discutir sobre programas disponíveis para encontrar equilíbrios de Nash; e (ii) em segundo lugar, com base nas propriedades teóricas de um equilíbrio de Nash, desenvolver um programa capaz de encontrar todos os equilíbrios de Nash puros em jogos com “n” jogadores e “m” estratégias (“n” e “m” sendo números finitos) como uma ferramenta Macro para o Microsoft Excel®. Espera-se que o programa possa contribuir para a área de Pesquisa Operacional, fornecendo uma nova ferramenta que facilite o uso dos conceitos da teoria dos jogos dentro dos cenários de resolução de problemas em grupo permitindo aplicações práticas usando um software difundido.

**Palavras-chave:** Tomada de decisão em grupo; Teoria dos jogos; Equilíbrio de Nash; Software.

Received June 10, 2019 - Accepted Dec. 28, 2019

Financial support: Fundação de Apoio a Pesquisa do Estado de São Paulo (2013/19915-9); Conselho Nacional de Desenvolvimento Científico e Tecnológico (458511/2014-5).



This is an Open Access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

## 1 Introduction

According to Angeloni (2003), in order to increase the quality of a group decision-making process, it is convenient to consider the heterogeneous preferences of the decision-makers and to improve their communication process. The former condition enables to assess the problem through different perspectives, allowing different possibilities to solve the problem. The latter is important for these groups to work efficiently. Game theory has been presented as a tool for dealing and promoting both conditions, expanding the way in which group decision making problems are dealt with.

Luce & Raiffa (1957, p. 5) had already informally characterized game theory as a mathematical formulation for situations of conflict among several people, whom is required to make choices from a well-defined set of strategies. Recently, Sanfey (2007, p. 318) states that “game theory provides a useful foundation for the study of decisions in a social context”, while Parsons & Wooldridge (2002) assert that “game theory is a close relative of decision theory [...] that can be considered the study of games against nature, where nature is an opponent that does not seek to gain the best payoff, but rather acts randomly”.

Myerson (1996) states that Nash equilibrium is a powerful concept of solution in game theory, since it provides stable solutions for strategic interactions scenarios. Conversely, according to Garey & Johnson (1977), the search of Nash equilibrium is considered a non-polynomial computational problem, in which exponential behavior can be noticed. Therefore, there is the need of computational tool for finding the Nash equilibria in games involving more than two players and strategies, since manually applying algorithms for this search is an extremely time-consuming task. According to the Game Theory Society (2019), a program to find Nash equilibria is a useful tool for those who want to find the theoretical results of their game theoretical models and compare them with the experimental results.

Available programs for finding Nash equilibria include *Gambit*, aimed at expert users, *Game Theory Explorer*, whose differential is its graphical interface and *GamePlan*, able to find all Nash equilibria in games with perfect or imperfect information, among others. However, none of the available programs can be easily integrated within a spreadsheet environment, which would make it easier the application of game theory for a wide range of problem-solving context. In this context, the aims of this research are twofold: (i) firstly, it was to identify and to discuss about the available programs for finding Nash equilibria; and (ii) secondly, to present a program for finding all pure Nash equilibria in games with “n” players and “m” strategies (“n” and “m” being finite numbers) as a Macro tool for Microsoft Excel®, a widespread spreadsheet software.

With the same goal of that the Microsoft Excel Solver® reached by “introducing students to optimization” (Fylstra et al., 1998), it is expected that the practitioners from industry and academy can use this new tool that can facilitate the use of game theory concepts in group decision making scenarios.

## 2 Available software for finding Nash equilibria

Although there are several algorithms for finding Nash equilibria, applying them manually becomes a time-consuming task. In this way, a program is needed to quickly and perfectly apply the developed algorithms. In the literature, it can be found software for

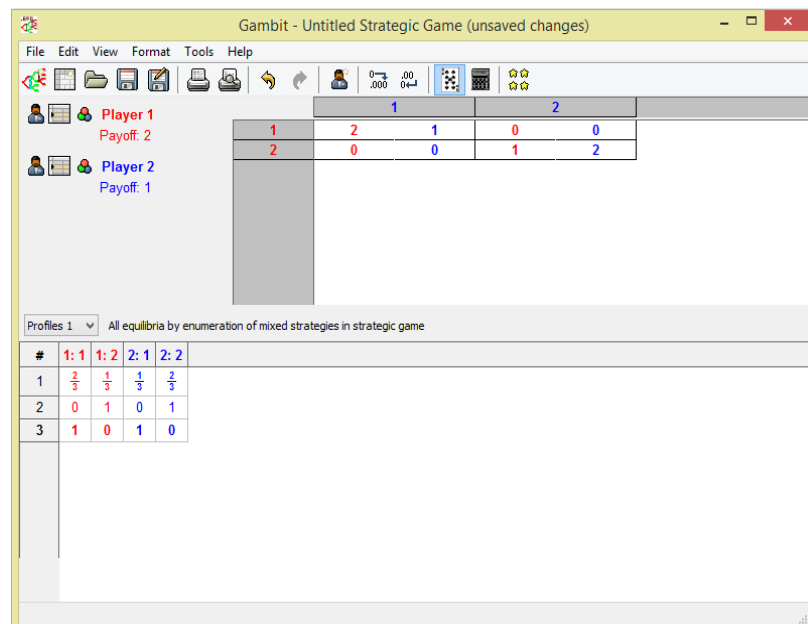
performing this task, including Gambit (McKelvey et al., 2006), Game Theory Explorer (GTE) (Egesdal et al., 2014) and GamePlan (Langlois, 2005), among others programs.

## 2.1 Gambit

Developed over 25 years, Gambit (<http://www.gambit-project.org/>) is an open-source software whose purpose is to find Nash equilibria. The programming language used in the last version was Python. The program can be used in Windows, Linux or Mac systems. Through Gambit, finite games can be constructed and analyzed extensively and strategically for non-cooperative games.

Gambit uses several algorithms to find Nash equilibria, presenting a command line interface for each of them (McKelvey et al., 2006). Among the command line interfaces, are: (i) Gambit-enumpoly, which finds Nash equilibria solving systems of polynomial equations; (ii) Gambit-enummixed, which solves 2-player games using the enumeration of extreme points; (iii) Gambit-gnm, which uses Newton's Global Method; (iv) Gambit-ipa, which uses the iterative method of multimatrix approximation; (v) Gambit-lcp, which solves 2-player games and uses the linear complementarity method; (vi) Gambit-lp, which solves 2-player games and finds equilibria through linear programming; (vii) Gambit-liap that uses a minimization function approach; and (viii) Gambit-simpdiv, which uses the subdivision approach.

Although this software has evolved significantly, Gambit depends on the inclusion of data through its interface (Figure 1), which might be a time-consuming task when the number of strategies or players increase. Alternatively, a Gambit's extension ".gbt" file could be considered for inputting the data. However, it should be noted that it depends on the user's programming abilities.



**Figure 1.** Gambit's screen with a game model for two players and two strategies and the results found by the software.

## 2.2 Game Theory Explorer

Game Theory Explorer, GTE (<http://www.gametheoryexplorer.org>) was developed with the objective of being integrated into the Gambit modules, being an open-source software in which, through a graphical interface for web browser, it enables extensive strategic iterative construction of games and ways for finding the equilibria for them (Egesdal et al., 2014). GTE focuses in providing a user-friendly interface for the use of non-specialists in such a way that visualization of the games becomes intuitive. Graphic shapes, such as tree-shaped games, can be customized, for example, vertically and/or horizontally. Games can be recorded as image format for later use in presentations. Within GTE it is possible to find all Nash equilibria for two players. On the other hand, it is noteworthy that the program's processing grows exponentially with the increase in the number of possible strategies. Thus, the number of strategies in GTE is restricted to around 15 to 20 per player. The program was written in ActionScript and JavaScript language and can be accessed via web. The Figure 2 presents the main screen of the software.

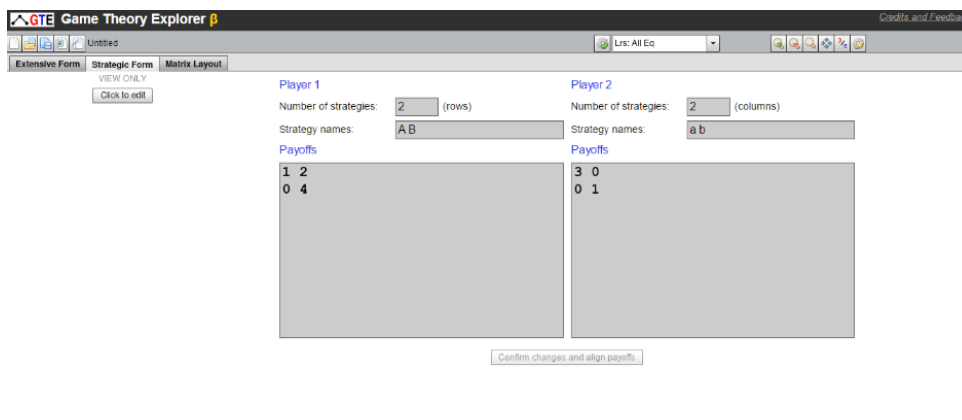


Figure 2. GTE's interface for a strategic 2-player game.

## 2.3 GamePlan

GamePlan (<http://userwww.sfsu.edu/langlois/>) was developed to create and solve a wide range of games in normal and/or extensive forms, with perfect or imperfect information, and static or repetitive games (Langlois, 2005). According to Langlois (2005) the development of GamePlan sought to achieve four objectives. The first was to be a friendly software, that is, it should be easy to create, edit and solve the games. Thus, the program features various colors to differentiate the players, payments and other information in order to facilitate distinguishing the data. The second goal was to be flexible. Thus, the program was sought to be capable of supporting virtually every type of game structure, limited only by the computer's memory capacity and speed. The third goal was to be exhaustive in relation to the possibilities of algorithms for calculating Nash equilibria. Finally, the fourth objective was to be fast. Figure 3 shows the software main screen in constructing a tree-shaped decision 3-player game, with two strategies each.

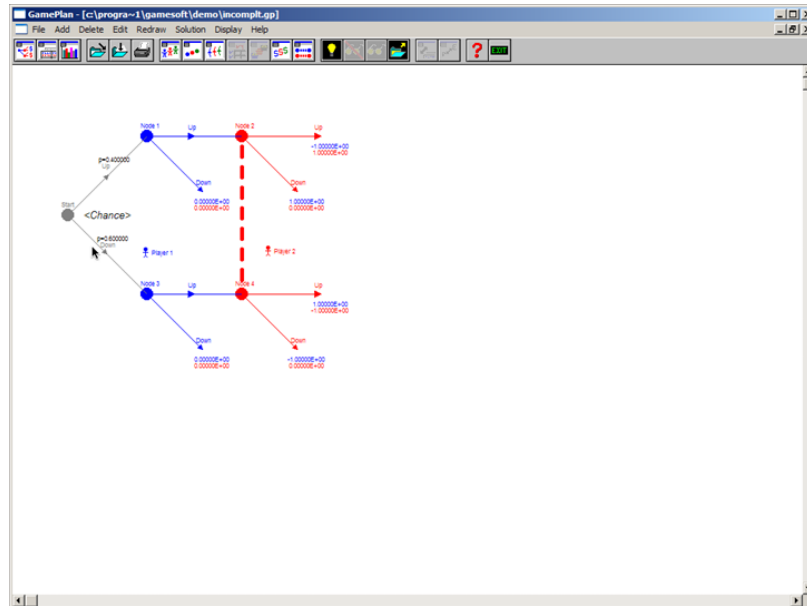


Figure 3. GamePlan's main screen of the software showing an extensive game model.

## 2.4 Other approaches

Dickhaut & Kaplan (1993) programmed the algorithm called “Nash.m” with the purpose of finding Nash equilibria in bimatrix games. The program converts any normal game into a symmetric game by creating equivalent solutions that are unlikely to be played, and search for pure and mixed Nash equilibria for two person games with a finite number of strategies. According to the authors, the program is useful for beginners to the process of finding Nash equilibria. However, because the program was designed to run with only two players, there is a clearly limitation on the use for both research and real cases. Regarding performance, the program also has the common feature with some other programs, that is, the time needed to find the equilibria grows exponentially in relation to the number of strategies per player.

Knight & Campbell (2018) developed a Python library for the computation of equilibria of two player strategic games called “Nashpy”. The library includes three algorithms for finding Nash equilibria, namely support enumeration, vertex enumeration, and Lemke Howson algorithm, as proposed in Nisan et al. (2007). According to Knight & Campbell (2018), Nashpy is simple to install, which is an alternative to Gambit for games with up to two players, since Gambit might be of difficult installation and, according to the authors, it is not portable. In addition, the implementation of Nashpy was designed in such a way to reduce the complexity of the algorithms and to bring the results with greater speed by not necessary finding all the equilibria.

Spaniel (2014) proposed the Game Theory Calculator, which is a Microsoft Excel® spreadsheet that search for pure strategy and mixed strategy Nash equilibrium for bimatrix games. However, although it's a practical environment of application, the spreadsheet is limited in number of players and strategies.

Krawczyk & Zuccollo (2006) presented a MATLAB package for finding Nash equilibria in finite games with n-players and m-strategies. However, this package was developed for specific game types. The MATLAB package only finds single equilibrium within the games. As the authors state, games that bring only a single equilibrium are important in cases of

regulatory economics and management. However, it is the same limitation as Nash.m, that is, the difficult to apply the program in research with a variety of real cases.

Finally, there are other contributions to the problem of finding Nash equilibrium that can be found in the literature. It can be cited the contributions of Lemke & Howson (1964), Herings & Peeters (2001), Govindan & Wilson (2003, 2004), Echenique (2007), and Porter et al. (2008). However, the mentioned studies provided only the algorithm for the search of the Nash equilibrium, which need to be programmed in order to be applied.

### 3 Methodology

Given the fact that available programs might be difficult to install or to make it portable and that they are either for specific uses or have the input of values as a tiring and time-consuming task it is proposed here a program for finding all pure Nash equilibria in non-cooperative games with “n” players and “m” strategies (“n” and “m” finite numbers) as a Macro to Microsoft Excel® using Visual Basic for Applications (VBA) programming language. The program was named Nash Equilibria Finder – NEFinder

The algorithm that supports the search of Nash equilibria in the NEFinder is a trivial method that exhaustively verifies all possible arrangements of strategies that satisfies the theoretical proprieties of the Nash equilibrium solution. The flowchart in Figure 4 summarizes the logical of the proposed algorithm.

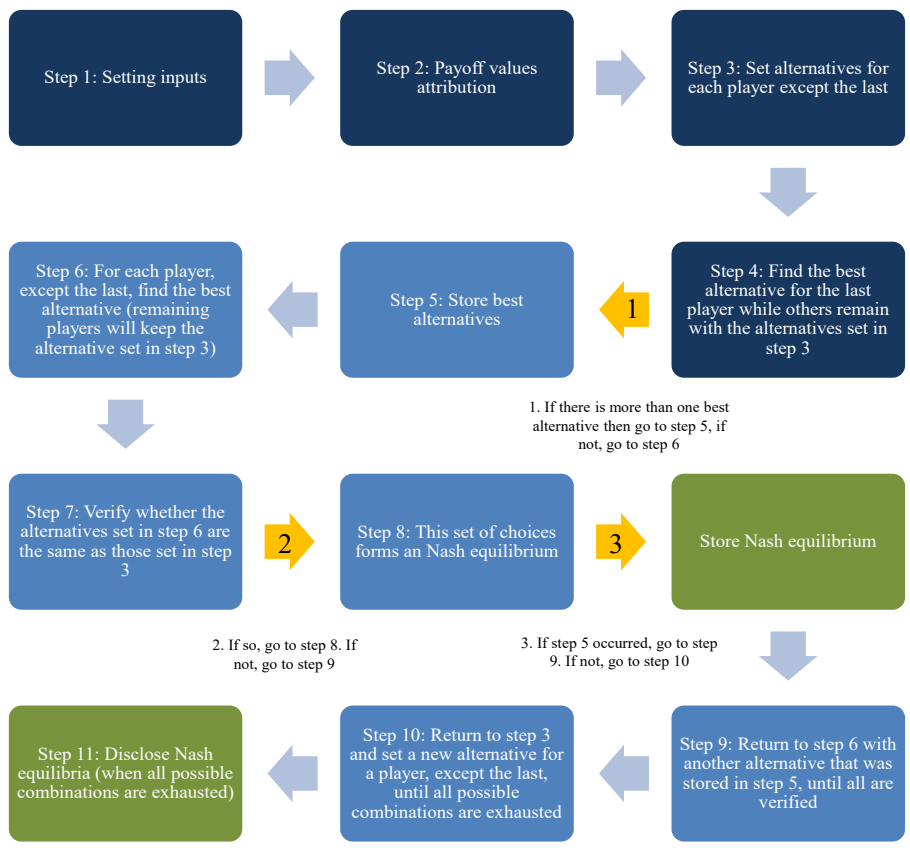


Figure 4. Flowchart of the algorithm within NEFinder.

Briefly, the procedure begins by defining the variables that will receive the input values, i.e.: number of players, number of possible strategies (alternatives), and matrix of payoffs of all the players, etc. In step 2, the payoff values are assigned to all players and to all strategies (this can be done very easily in a spreadsheet environment). In step 3, an arrangement of strategies must be fixed for all players except one, since this player will be free to choose the best strategy based on the strategies set to the other players. Therefore, in step 4, the strategy with highest payoff for this player will be searched, considering the arrangement of strategies previously fixed to the other players. If more than one highest payoff to this player is found, these other strategies with the same payoff must be stored. This is done in step 5. From step 6 to step 8, the procedure for finding Nash equilibria is run based on the theoretical properties of the Nash equilibrium solution. Specifically, a strategy arrangement  $s^*$  will be a Nash equilibrium of a strategic game with “n” players if, and only if, for every player  $i$  and an arbitrary strategy  $s_i$ ,  $u_i(s_1^*, s_2^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_n^*) \geq u_i(s_1^*, s_2^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_n^*)$ , where  $u_i$  is the utility of player  $i$  (Osborne & Rubinstein, 1994). In other words, a strategy arrangement  $s^*$  will be a Nash equilibrium of a strategic game if, and only if, every player’s strategy  $s_i^*$  is a best response to the other players’ strategies  $(s_1^*, s_2^*, \dots, s_{i-1}^*, s_{i+1}^*, \dots, s_n^*)$ . In this way, all possible arrangements of choices for the players should be verified. In step 6, each player, except the one that was firstly set to having his choice free, will choose their best strategies, but with one restriction imposed, namely: the player  $i$  must choose a strategy while other players remain with their strategies fixed. In step 7, it is verified whether the strategies chosen in step 6 are the same as those set in the step 3. If so, a Nash equilibrium was found and it is stored (step 8). Otherwise: (i) if more than one strategy has been stored in step 5, the procedures go back to step 6 and check all them (step 9); or (ii) if no strategy have been stored in step 5, then the procedure goes back to the step 3 and set another arrangement of strategies to the players, except the one that was firstly set to having his choice free, until all possible arrangements of strategies have been verified. The algorithm is detailed in the pseudocode bellow.

---

### Pseudo-code for exhaustively find all pure Nash equilibria within NEFinder

---

```

Begin
  Step (1):
    set number of player = nplayer
    set number of payoffs per player = npay
    set payments = payments(0 to npay-1, 0 to nplayer-1)
    set chosen alternative = chosenalternative (0 to nplayer-1)
    set alternatives= alternatives (0 to number of alternatives - 1)

  Step (2)
    For n = 0 to nplayer - 1
      For i = 0 to njog - 1
        payments(i,n) = value of i payment for n player
      next i
    Next n

  Step (3)
    For n = 0 to nplaer - 2
      chosenalternative (n) = set one alternative for each n
    Next n

```

Step (4)

n = nplayer - 1

downpayment = 0

**For** i = 0 to nplayer - 1

**If** payments (i, n) > downpayments **Then**

**For** m = 0 to nplayer - 2

**If** "chosenalternative (m) <> fixed alternative  
            plyer m" **Then**

                m = nplayer

**End if**

**Next m**

**If** m = nplayer - 2 **Then**

            chosenalternative (n) = alternative (i)

            downpayments = payments(i, n)

**End if**

**End if**

**Next i**

Step (5)

**If** two or more alternatives exist that satisfy the above condition **Then**  
    should store them

**End if**

Step (6)

**For** i = 0 until nplayer - 1

    For the player i check which alternative is chosen in the case of  
    the other choose the alternative fixed and the player n choose the  
    "chosenalternative"

**Next i**

Step (7)

**If** "all players choose the previously established alternatives" **Then**

    The set of choices form a equilibrium

**End if**

Setp (8)

**If** in step(5) there is more than one option **Then**

    repeat the step (6) and (7) for all options

**End if**

Step (9)

**If** all possible alternatives have been fixed **Then**

    end

**Else**

        back to the step (3) and fix other alternative for a different player

**End if**

Output (found equilibria)

End

---

To illustrate the logic of the algorithm, a game with three players and two strategies is presented here. All possible arrangements for a game with three players and two strategies are presented in Table 1. In the first time that the algorithm arrives at step 3 the strategy "A" will be set for all players except to the last. In the next iterations, the algorithm will set some other possible arrangement described, for example, the stragtegy "B" for Player 1 and

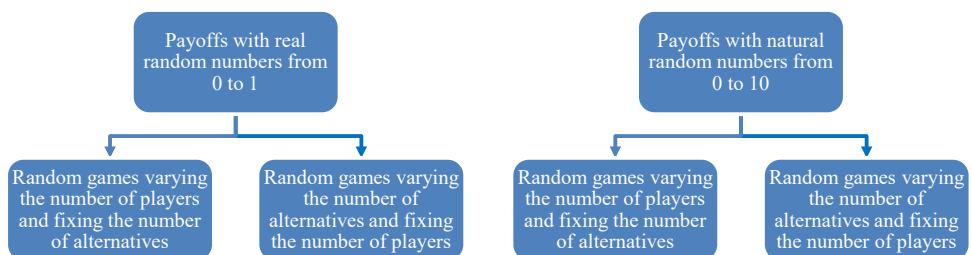


strategy “A” for Player 2. At each iteration a new arrangement must be chosen to set the strategies to the players until all possible arrangements have been verified.

**Table 1.** Possible arrangements for a game with 3 players and 2 strategies.

	Player 1	Player 2	Player 3
Arrangement of possible strategies that can be set	A	A	Free
	B	A	Free
	A	B	Free
	B	B	Free

To calibrate the NEFinder, tests with random games were performed. As can be seen in Figure 5, tests were performed with payoffs being random real numbers ranging from 0 to 1, rounding to four decimal places, and the same tests were performed with the payoffs as natural random numbers ranging from 0 to 10, since it was necessary to calculate the efficiency of the NEFinder for situations where there is more chance to find strict Nash equilibria (in the tests with natural numbers) than to those where few strict Nash equilibria are possible (in the tests with real numbers). Tests were also performed by varying the number of players and keeping the number of strategies constant and then varying the number of strategies and keeping the number of players constant. For calculating the processing time of NEFinder, the computer screen was recorded in the Camtasia Studio 8 (video editor software) with the program running together with the Windows chronometer and, afterward, the video was analyzed by slow-motion play for verifying the time of begin and ending of process for calculating the overall time consumed in each test. All tests were performed using an Intel i7-4790 CPU with 3.60 Ghz and 16 GB RAM in a Windows 10 Pro environment. The Microsoft Excel® version was the one from Microsoft Office Professional Plus 2013.



**Figure 5.** Tests performed to analyze and validate *NEFinder*.

Finally, in order to validate the calculations of NEFinder, Gambit software was used for comparisons and verification.

## 4 Results and discussion

Some of the main differences between the available programs in the literature are related to the handling of the software and to their interface. For instance, while for Gambit it is possible to download the software, GTE must be used online. This means that, in order to use Gambit, certain software installation technical experience is necessary. On the other

hand, GTE is more user-friendly, since the focus of the program was precisely to be easily handled by the users. In its turn, GamePlan can be downloaded and installed in a desktop.

Particularly, none of these programs can be used together with Microsoft Excel®, which is a very disseminate tool, commonly used by analysts. Furthermore, the process for including payoffs is manual and it is on the basis of one per time, which makes the integration of the programs with other applications very difficult. Therefore, the main innovation of NEFinder is the searching of Nash equilibria within a spreadsheet environment. The NEFinder program was written in Visual Basic for Application (VBA) language. The VBA language was chosen to be possible to execute NEFinder as a Macro of Microsoft Excel®.

For using NEFinder in the Microsoft Excel® environment, it is necessary, previously of opening the spreadsheet that is going to be manipulated, to open the Macro containing the algorithm of NEFinder, which is the file “NEFinderV1P.xlam” for the Portuguese version of the program or the file “NEFinderV1E.xlam” for the English version of the program. The NEFinder algorithm then will create a menu in the Microsoft Excel® add-in tab containing three menu buttons. The first leads to the presentation of the program, as can be seen in the Figure 6.

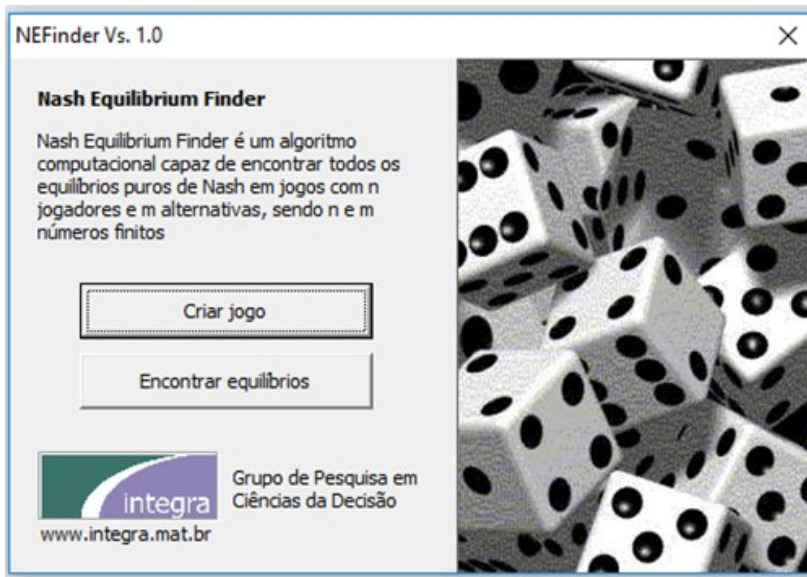


Figure 6. NEFinder's starting screen.

The second opens the window related to the creation of the game. Through this window it is possible to set the number of players, strategies and the name of the new spreadsheet where the game will be created. The NEFinder demands the number of players and strategies to be at least two, as the bimatrix format is the basic setting for running game theory frameworks. There is no upper limit on the number of players or alternatives, although the computational nature of the problem might restrict it significantly in terms of processing time and, therefore, should be considerate by the user. Additionally, Microsoft Excel® has a limit of up to a million rows, which should be also considered as a limitation, since the number of rows demanded by NEFinder will be the same of the possible arrangements given by the equation  $arrangements = m^n$ , where “n” is the number of players and “m” is the number of strategies (alternatives). Figure 7 shows NEFinder's game creation screen.

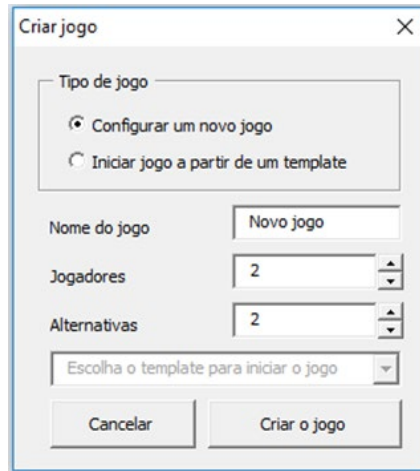


Figure 7. NEFinder's starting screen for game parameters.

After setting the number of players and strategies, a new game can be created in the strategic form by clicking “Criar o jogo”. Each player is represented by a column and all arrangements of strategy are shown in the rows. Figure 8 shows the configuration for a two-players game with two strategies each. The columns on the left side of the gray square shows the possible strategies with different colors, each color representing the strategy of one player. For example, row 4 contains the arrangement of strategies in which both players opt for strategy “1”, while in row 5, Player 1 (red) chooses strategy “2” and Player 2 (green) chooses strategy “1”. The colors for each player are automatically defined by the program to facilitate visualization for the user.

	A	B	C	D	E
1	Jogo para 2 jogadores 2 alternativas.				
2					
3			jogador 1	jogador 2	
4	1	1			
5	2	1			
6	1	2			
7	2	2			
8					
9					

Figure 8. Template of 2-player game with 2 strategies.

The cells within the grey square will receive the payoff values for each player associated to each arrangement of strategies, with a validation that prevents the user from entering texts instead numbers. For example, if a row contains the number two in red and the number one in green it means that the Player 1 has chosen strategy “2” and the Player 2 has chosen strategy “1”. Suppose that the payoff for Player 1, in the event that both players choose strategy one, is “two”, then where the Player 1 column intersects with the row for the corresponding strategy the cell will receive the value equal to “two”. Suppose also that when both players choose strategy one, the payoff for Player 2 is “one”. Thus, in the same row, but in the column related to Player 2 it is inserted the value equal to “one”. Accordingly, the payoff value for Player 1 should be inserted into cell C4, while the payoff value for Player 2 should be inserted into cell D4, when both choose strategy “1”

as their strategies (row 4). The same logic is applied until all the payoffs are inserted into the grey cells. Figure 9 contains these and other arbitrary values for illustration.

	A	B	C	D	E
1	Jogo para 2 jogadores 2 alternativas.				
2					
3			jogador 1	jogador 2	
4	1	1	2	1	
5	2	1	0	0	
6	1	2	0	0	
7	2	2	1	2	
8					
9					

Figure 9. Example of player's payoffs.

The third button opens the window of NEFinder that is concerned with the calculation of equilibria (Figure 10). One need only to select the grey area in which the payoffs were entered and to click at "Encontrar equilíbrios" for finding all pure Nash equilibria based on the payoffs values. After selecting the payoffs and clicking "Encontrar equilíbrios", the program will run the trivial and exhaustive searching algorithm and the Nash equilibria found (if any) and their respective payoffs are displayed for each equilibrium on the same page in which the game was created (Figure 11). One can verify that the Nash equilibria found to the game presented in the Figure 9 would be when the Player 1 chooses the strategy "1" as the best response to the choice of Player 2 for strategy "1" and when the Player 1 chooses the strategy "2" as the best response to the choice of Player 2 for strategy "2", since  $u_1(1,1) > u_1(2,1)$ , and  $u_1(1,2) < u_1(2,2)$ , and the same logic to the Player 2.

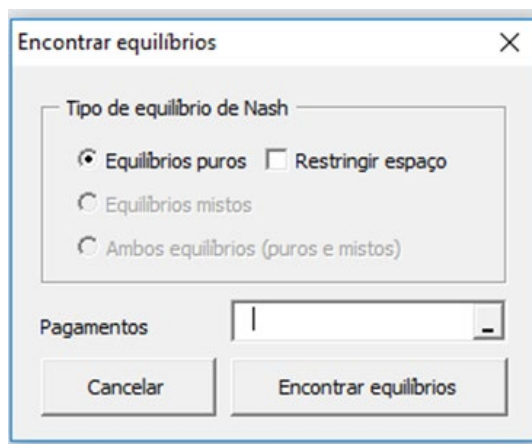


Figure 10. NEFinder's payoff selection screen.

	A	B	C	D	E	F	G	H
1	Jogo para 2 jogadores 2 alternativas.							
2								
3			jogador 1	jogador 2				
4	1	1	2	1				
5	2	1	0	0				
6	1	2	0	0				
7	2	2	1	2				
8								
9			jogador1	jogador2			jogador1	jogador2
10		equilibrio 1	1	1		pagamentos	2	1
11		equilibrio 2	2	2		pagamentos	1	2
12								

Figure 11. Pure Nash equilibria found and player's payoffs for each equilibrium.

Finally, random simulations were performed for evaluating the performance of NEFinder with natural and real numbers. The average processing time was evaluated firstly by maintaining the number of strategies fixed and varying the number of players. Subsequently, average processing time was also measured, however, varying the number of strategies and keeping the number of players constant. Figure 12 shows the average times obtained with the games in which the number of strategies were fixed, varying the number of players, using randomly payoffs with positive natural numbers from 0 up to 10. The blue line represents the games with two strategies where the numbers of players varied from four players up to nine. The red line represents average times obtained in the simulations of games with three strategies, with players varying between four and seven. Finally, the same procedure was conducted for games with four strategies in which the number of players varies from four to six (green line). In the second stage of the tests with natural numbers, the number of players was kept constant and the number of strategies varying. Figure 13 shows the average processing time elapsed to find Nash equilibria in games with four players, varying the number of strategies from three to six (blue line) and the average processing time of games for five players by varying the number of strategies, also from three to five (red line).

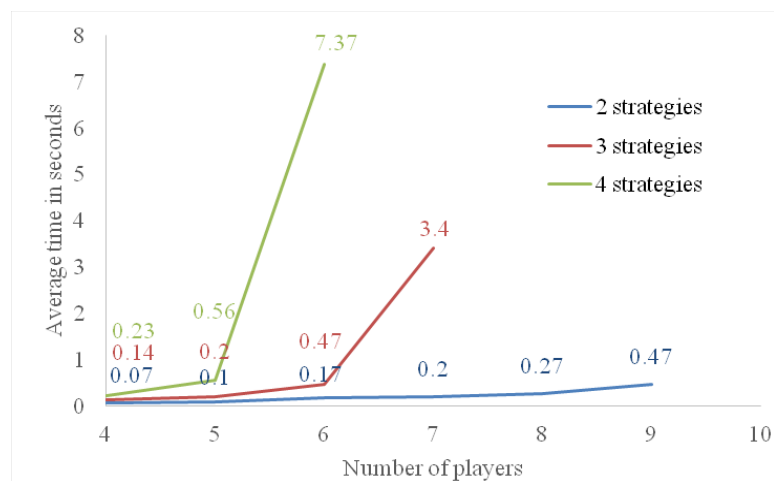
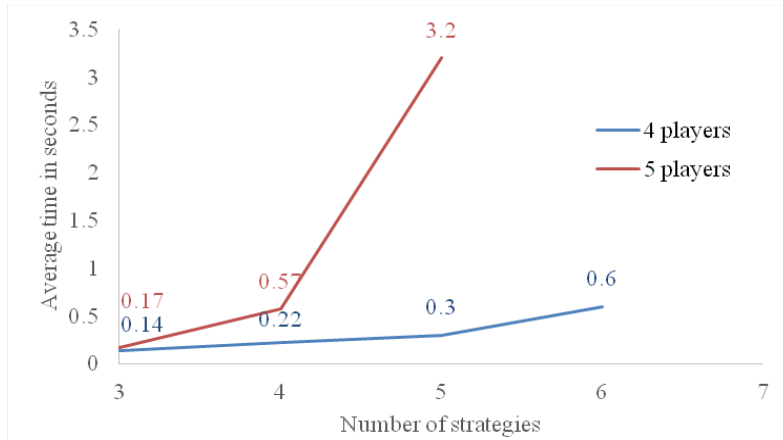


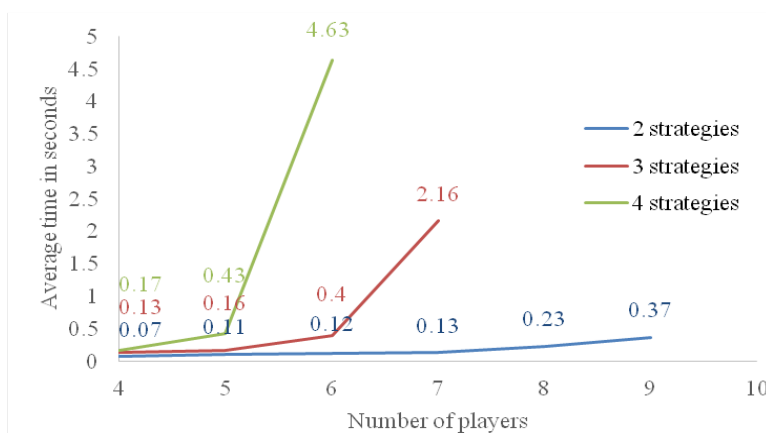
Figure 12. Time elapsed varying the number of players for games with natural numbers.



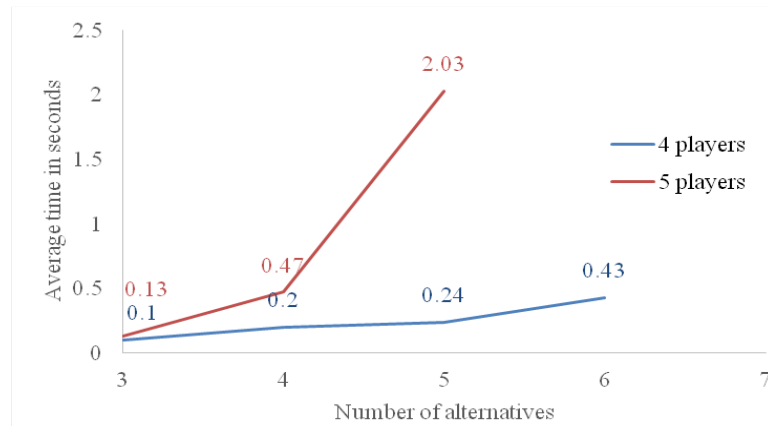
**Figure 13.** Time elapsed varying the number of strategies for games with natural numbers.

As expected by the nature of the problem (Garey & Johnson, 1977), in both first and second sets of tests exponential behavior was noticed and can be seen, in most of cases, in the graph, although some are not noticeable by the limitation of the number of strategies used. The exponential behavior was expected due to the fact that the algorithm used for NEFinder checks all possible arrangements of choices, verifying if each of the arrangement forms a Nash equilibrium. Thus, the processing time increases exponentially with the increase in the number of either the number of players or the number of strategies.

The same performance tests were repeated, but this time with the values of the payoffs being real numbers varying randomly from 0 up to 1. The Figure 14 presents average processing times for the games varying the number of players from four to nine and keeping the number of strategies fixed at two (blue line), varying the number of players from four to seven and keeping the number of strategies fixed at three (red line) and varying the number of players from four to six and keeping the number of strategies fixed at four (green line). In the second stage of the tests the number of players was kept fixed and the number of strategies varied. Tests were performed with four players and varying the number of strategies from three to five and with five players (red line), also varying the number of strategies from three to six with four players (blue line). The average processing time of these tests are shown in the Figure 15.



**Figure 14.** Time elapsed varying the number of players for games with real numbers.



**Figure 15.** Time elapsed varying the number of strategies for games with real numbers.

It can be noted that NEFinder speed is always faster when using real numbers. It should be explained by the fact that the number of required loops that NEFinder algorithm run for situations where too many non-strict Nash equilibria are present might be high, as the case for the games with natural numbers. Nevertheless, it should be noted that, for natural or real numbers, the performance of games with two players and two strategies were always very fast, meaning that NEFinder is very suitable for the majority of situations modeled by the use of game theory approach.

For validating the results found in NEFinder, the payoffs that were randomly generated in the Microsoft Excel® spreadsheet were converted into the “.GBT” format, which is the format of files manipulated by Gambit. All the tests were performed by using the Gambit’s algorithm to compute as many Nash equilibria as possible by looking for pure strategy equilibria, which has similar purpose of NEFinder. The comparison analysis of NEFinder and Gambit showed perfect convergence of results.

## 5 Conclusions

Through a literature review, it was possible to identify available programs aimed at finding Nash equilibria. Here is proposed another option for practitioners, the NEFinder. In relation to comprehensive programs, such as Gambit or GamePlan, NEFinder is clearly disadvantageous with relation to possible results that can be found (pure and mixed equilibria). On the other hand, NEFinder has advantages, including the possibility of using a spreadsheet environment without a preliminary setup phase and the possibility of allowing several games simulations without the need to reinsert data, which can be automatized through other Microsoft Excel® Macros. It is also noted that NEFinder is able to find Nash equilibria for games with “n” players and “m” strategies, which is an advantage over programs such as Game Theory Explorer, Nash.m, Nashpy, and Game Theory Calculator. Finally, in relation to speed of calculation, NEFinder demonstrated to be faster when manipulating real numbers between zero and the unity with less probability of the presence of non-strict Nash equilibria. Due to the fact that NEFinder does not require a preliminary learning phase for using the program, it can increase the efficiency of the program on the overall time for finding Nash equilibria.

Therefore, the main advantage of NEFinder is due to the fact that it can be used in Microsoft Excel® and makes filling up the payoff values very quick, besides Microsoft Excel® being widely used and well known to users. Another advantage of NEFinder is

related to the simulation of different games that can be created using different spreadsheets in the same file, which make it possible to run sensitivity analysis for evaluating the results. This feature makes possible the use of NEFinder for instruction of many most common games, including Prisoner's Dilemma, Stag Hunt game, Chicken game, Battle of the Sexes, and, specially, coordination games. Therefore, NEFinder's contribution is not based on the complexity of the program, but on its innovative application of easy visualization and practicality in handling. It is expected that NEFinder can contribute to the dissemination of game theory by serving as a tool for finding pure Nash equilibria in group decision making, as before Microsoft Excel Solver® has done in optimization area.

It is recommended to use NEFinder rather than other program in cases where it is desirable to obtain only pure equilibria. It is generally an important feature in many decision-making cases. When it is necessary to find mixed equilibria, it is suggested to use other programs, since the first version of NEFinder does not have this function.

## References

- Angeloni, M. T. (2003). Elementos intervenientes na tomada de decisão. *Ciência da Informação*, 32(1), 17-22. <http://dx.doi.org/10.1590/S0100-19652003000100002>.
- Dickhaut, J., & Kaplan, T. (1993). A program for finding Nash equilibria. *The Mathematica Journal*, 1, 87-93.
- Egesdal, M., Jordana, A. G., Prause, M., Savani, R., & Stengel, B. V. (2014). *Game Theory Explorer*. Retrieved in 2014, October 15, from <http://www.gametheoryexplorer.org/>
- Echenique, F. (2007). Finding all equilibria in games of strategic complements. *Journal of Economic Theory*, 135(1), 514-532. <http://dx.doi.org/10.1016/j.jet.2006.06.001>.
- Fylstra, D., Lasdon, L., Watson, J., & Waren, A. (1998). Design and use of the Microsoft Excel Solver. *Interfaces*, 28(5), 29-55. <http://dx.doi.org/10.1287/inte.28.5.29>.
- Game Theory Society. (2019). Retrieved in 2014, October 15, from <http://www.gametheorysociety.org/>
- Garey, M. R., & Johnson, D. S. (1977). Computers and intractability: A guide to the theory of NP-completeness. *The Freeman*
- Govindan, S., & Wilson, R. (2003). A global Newton method to compute Nash equilibria. *Journal of Economic Theory*, 110(1), 65-86. [http://dx.doi.org/10.1016/S0022-0531\(03\)00005-X](http://dx.doi.org/10.1016/S0022-0531(03)00005-X).
- Govindan, S., & Wilson, R. (2004). Computing Nash equilibria by iterated polymatrix approximation. *Journal of Economic Dynamics and Control*, 28(7), 1229-1241. [http://dx.doi.org/10.1016/S0165-1889\(03\)00108-8](http://dx.doi.org/10.1016/S0165-1889(03)00108-8).
- Herings, P. J. J., & Peeters, R. J. (2001). A differentiable homotopy to compute Nash equilibria of n-person games. *Economic Theory*, 18(1), 159-185. <http://dx.doi.org/10.1007/PL00004129>.
- Krawczyk, J., & Zuccollo, J. (2006). *NIRA-3: An improved MATLAB package for finding Nash equilibria in infinite games*. Victoria University of Wellington.
- Knight, V. A., & Campbell, J. (2018). Nashpy: A Python library for the computation of Nash equilibria. *Journal Open Source Software*, 3(30), 904. <http://dx.doi.org/10.21105/joss.00904>.
- Lemke, C., & Howson, J. Jr (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2), 413-423. <http://dx.doi.org/10.1137/0112033>.
- Luce, R. D., & Raiffa, H. (1957). *Games and decisions: Introduction and critical survey*. New York: Wiley.
- McKelvey, R. D., McLennan, A. M., & Turocy, T. L. (2006). *Gambit: Software tools for game theory*. Dover Publications.



- Langlois, J. P. (2005) *An introduction to Game Theory Using the GamePlan Software*. Retrieved in 2014, October 15, from [https://www.researchgate.net/profile/Jean-Pierre\\_Langlois/publication/238098439\\_An\\_Introduction\\_to\\_Game\\_Theory\\_Using\\_the\\_GamePlan\\_Software/links/02e7e52d5a9cc8d3ac000000.pdf](https://www.researchgate.net/profile/Jean-Pierre_Langlois/publication/238098439_An_Introduction_to_Game_Theory_Using_the_GamePlan_Software/links/02e7e52d5a9cc8d3ac000000.pdf)
- Myerson, R. (1996). John Nash's Contribution to Economics. *Games and Economic Behavior*, 14(2), 287-295. <http://dx.doi.org/10.1006/game.1996.0053>.
- Nisan, N., Roughgarden, T., Tardos, E., & Vazirani, V. V. (2007). *Algorithmic game theory* (vol. 1). Cambridge: Cambridge University Press. <http://dx.doi.org/10.1017/CBO9780511800481>.
- Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. Cambridge: MIT press.
- Parsons, S., & Wooldridge, M. (2002). Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3), 243-254. <http://dx.doi.org/10.1023/A:1015575522401>.
- Porter, R., Nudelman, E., & Shoham, Y. (2008). Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2), 642-662. <http://dx.doi.org/10.1016/j.geb.2006.03.015>.
- Sanfey, A. G. (2007). Social decision-making: insights from game theory and neuroscience. *Science*, 318(5850), 598-602. <http://dx.doi.org/10.1126/science.1142996>. PMID:17962552.
- Spaniel, W. (2014). *Game theory 101: the complete textbook*. CreateSpace.