I.J. Mathematical Sciences and Computing, **2016, 3, 32-40**
Published Online July 2016 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2016.03.03

***Available online at** http://www.mecs-press.net/ijmsc*

Quantum Walk Algorithm to Compute Subgame Perfect Equilibrium in Finite Two-player Sequential Games

Arish Pitchai [a,*], A V Reddy [a], Nickolas Savarimuthu [a]

[a] *Dept. of Computer Applications, National Institute of Technology Tiruchirappalli, Trichy 620015, India*

Abstract

Subgame Perfect Equilibrium (SGPE) is a refined version of Nash equilibrium used in games of sequential nature. Computational complexity of classical approaches to compute SGPE grows exponentially with the increase in height of the game tree. In this paper, we present a quantum algorithm based on discrete-time quantum walk to compute Subgame Perfect Equilibrium (SGPE) in a finite two-player sequential game. A full-width game tree of average branching factor *b* and height *h* has $n = O(b^h)$ nodes in it. The proposed algorithm uses $O(n/b)$ oracle queries to backtrack to the solution. The resultant speed-up is $O(b)$ times better than the best known classical approach, Zermelo's algorithm.

Index Terms: Quantum Game Theory, Sequential Games, Subgame Perfect Equilibrium, Quantum Random Walk, Backward Induction, Two-player Games, Quantum Algorithm, Quantum Computing.

1. Introduction

Game theory is the branch of mathematics that deals with strategic decision making in conflict situations. The two frequently used representations to describe a game are the normal form and the extensive form. Normal form represents the game with the help of a matrix. Extensive form is the dynamic description of a game where the sequential actions of players are modeled in detail with a directed tree. Nash equilibrium is a solution concept used to predict the outcome of the strategic interaction. To avoid misleading predictions and to protect its credibility in certain circumstances, Nash equilibrium has undergone multiple refinements. In extensive form games, Nash equilibrium fails to consider the sequential structure, therefore a related solution concept called Subgame Perfect Equilibrium for dynamic games was proposed by Reinhard Selten [1].

Zermelo's backward induction is the predominant classical solution concept, used to find equilibrium points in extensive form games [2]. Even though Zermelo's algorithm is known to solve game tree of size *n* with *O(n)*

* Corresponding author. Tel.: +91-9677376606
E-mail address: arish.p@gmail.com

queries, the number of queries is enormous for large game trees. In this paper, we present a quantum algorithm that needs *O(n/b)* queries to find the solution (*b* is the average branching factor), which considerably reduces the execution time. Our algorithm employs quantum walk, an advanced search tool for building quantum algorithms. Quantum walks are the quantum analogue of classical random walks. Harnessing the exponential speed-up in their hitting time has resulted in the development of many quantum algorithms that outperform their classical counterparts [3].

The two types of quantum walks, based on their behavior, are discrete-time and continuous-time quantum random walks. The proposed algorithm is a combination of backward induction and perturbed discrete-time quantum walk. Given game tree is broken down into smallest possible sub-trees starting from the lowest level. Resultant sub-trees are then transformed into hypercube search spaces one by one. Application of perturbed discrete quantum walk to each sub-tree identifies equilibrium node in that sub-tree. Finally, backward dynamic programming approach is applied to compute the subgame perfectness. For simplicity, the algorithm is assumed to be applied on a full-width game tree with a uniform branching factor and an even numbered height. If the oracular coin is capable of identifying equilibrium points of *k* players, then our algorithm can solve this problem with $O(n/b^{(k-1)})$ queries.

The paper is organized as follows. We begin section 2 with relevant definitions of game tree, Nash equilibrium and subgame perfect equilibrium. Reviewing the classical available techniques to solve the problem is done in section 3. Quantum algorithm that identifies subgame perfect equilibrium is described in section 4. In section 5, results of applying the algorithm to a two-player game and its generalization is discussed in detail. We conclude this paper in section 6 with the limitations of our algorithm and open problems in quantum game theory.

2. Problem Definition

A detailed dynamic description of a game where the sequential actions of players are modeled in detail with a directed tree, is the extensive form representation. Subgame perfect equilibrium (SGPE) is the equilibrium refinement of Nash equilibrium in sequential games. The given problem of computing SGPE in a finite two-player sequential game is formally defined in this section.

Definition 1. A *two-player finite sequential game tree* is given by a tuple $T = (\{X_i\}_{i \in h}, q)$ where

- h is the height of the game tree.
- X_i is the set of moves at level i, where odd and even levels are played by *player-1* and *player-2* respectively.
- $q: X_1 \times \ldots \times X_h \rightarrow \mathrm{R}^2$ is the outcome function.

A play in the game is a tuple $\vec{x} = (x_1, \ldots x_h)$. For each play $\vec{x}$, the 2-tuple $q(\vec{x})$ describes the payoffs obtained by each of the two players in that run of the game.

Definition 2. A *strategy* for the player playing i^{th} turn is a mapping

$$s_i : X_1 \times \ldots \times X_{i-1} \rightarrow X_i$$

describing what move the player should choose based on what has been played up to the point $i - 1$.

Definition 3. A *strategy profile* is a tuple of strategies $(s_i)_{i \in h}$.
A strategy profile determines a play $\vec{x}$ as

$$x_i = s_i(x_1, \ldots, x_{i-1})$$

and its corresponding outcome $u = q(\vec{x})$. We write q_j: $X_1 \times \ldots \times X_h \to \mathrm{R}$ for the payoff of the j^{th} player in q. Hence, given a play $\vec{x}$, the payoff of player j is given by $q_j(\vec{x})$. We also write $q_j(s_1, \ldots, s_h)$ for the j^{th} player's outcome, determined by the strategy profile $(s_i)_{i \in h}$.

Definition 4 (Nash equilibrium). A strategy profile $(s_i)_{i \in h}$ is in *Nash equilibrium* if for each player j and alternative strategy s_i^* we have

$$q_j(s_1, \ldots, s_i, \ldots, s_h) \geq q_j(s_1, \ldots, s_i^*, \ldots, s_h)$$

Informally, a strategy profile is in equilibrium if no player has an incentive to unilaterally change his strategy.

Definition 5 (Subgame). In any two-player sequential game $(\{X_i\}_{i \in h}, q)$ a partial play $x_1, \ldots, x_{i-1}$ determines a *subgame* $T_i = (\{X_k\}_{i \leq k \leq h}, \tilde{q})$ with payoff function

$$\tilde{q}_j(x_i, \ldots, x_h) = q_j(x_1, \ldots, x_{i-1}, x_i, \ldots, x_h)$$

for the completion of the play $(x_i, \ldots, x_h)$.

Definition 6 (Subgame perfect equilibrium). A strategy profile $(s_i)_{i \in h}$ is in *subgame perfect equilibrium* if it is in Nash equilibrium on any subgame.

Our problem is informally defined as the identification of such sequential strategy profile in a game tree, i.e., the Subgame perfect equilibrium.

3. Related Work

In this section, we review related work on Subgame perfectness in game trees. A brute force computation requires examining every state in the normal or the matrix form representation of the game. It takes exponential time to compute the subgame perfect equilibrium because pruning the search space is possible only in extensive form. Since it suffers from combinatorial explosion, generalization of zermelo's algorithm is preferred over brute force [4]. It reduces the computational complexity of the problem.

Zermelo's algorithm is the predominant classical approach used to compute equilibrium in extensive form games of complete information [5]. It uses backward dynamic procedure to solve the problem. Selection of action $x_i \in X_i$ at each level i of the game is based on the highest utility value of possible action, which is chosen from the lowest level to the highest. Mechanically, it is computed as follows. Consider any non-terminal node that comes just before terminal nodes, i.e., after each action branching from this node, the game terminates. Common sense of the player who moves at this node will make him choose the best move, the move that gives him the highest utility for himself. The sub-tree stemming from this node is deleted after assigning the payoff vector associated with current selection to the non-terminal node at hand, so that we have a shorter game, where the current non-terminal node is now a terminal node. This procedure is repeated until the root is reached [6]. Number of queries required by this algorithm is proportional to the size of the game tree ($O(n) = O(b^h)$, where n is the number of nodes in the game tree, b is its average branching factor and h is its height).

Even though backward induction provides an algorithm very much better than the brute force approach, it isn't good enough for some games that require sequential decisions. For example, game tree of a reasonable chess game comes with $b \approx 35$ and $h \approx 100$, backward induction needs $35^{100} \approx 10^{130}$ queries to identify the solution, which seems exponential in time [7]. Game tree complexity of well-known strategy games like chess, stratego and havannah [8] exceeds the total number of atoms in the universe ($\approx 10^{80}$). Recent work that addressed a similar problem in zero-sum games uses randomized techniques to bring the computational complexity down [9]. Game tree search space is reduced by guessing or sampling other player's moves and then the conventional approach is applied to obtain the solution. Even though application of random sampling, results in a computational complexity that is independent of the size of the game, probability of finding the solution depends mainly on the (random) selection of sub-matrices.

Pioneering works on quantum games prove that quantum strategies are always at least as good as classical strategies [10-12]. Using quantum entanglement as a useful tool, backward induction is shown to generate equilibrium out of the quantum form of Stackelberg duopoly [13]. Quantum walk, which evidently outperforms its existing classical algorithms, is used to design various algorithms for different search spaces [14-17]. Since its origin, quantum walk is an effective tool for searching graphs. Hence, we use it for the construction of our quantum algorithm in this paper.

4. Quantum Walk Algorithm

A linear time quantum algorithm based on discrete quantum walk is given in this section. We begin by describing discrete quantum walk model in section 4.1 and quantum walk search in section 4.2. Finally, we explain how the quantum walk search can be combined with backward induction to compute the subgame perfect equilibrium in section 4.3.

4.1. Discrete Quantum Walk Model

Our model, similar to that of the one used by Aaronson and Ambainis [18], is the quantum counterpart of classical random walk. The two basic components of this model are a Quantum walker and a Quantum coin. The walker moves in a Hilbert space (vertex space - $\mathcal{H}_v$) of infinite, but countable dimensions D. The outcomes of the coin flips, span the Hilbert space (coin space - $\mathcal{H}_c$) of arbitrary fixed dimension d. More specifically, a quantum walk is performed on a Hilbert space $\mathcal{H} = \mathcal{H}_c \otimes \mathcal{H}_v$ and state of the system is represented as $|\psi\rangle$ = $|\text{coin, position}\rangle$, where $|\text{coin}\rangle \in \mathcal{H}_c$ and $|\text{position}\rangle \in \mathcal{H}_v$.

Evolution of a quantum system in $\mathcal{H}$ is described by the unitary operator U = S. C. It is a two-step process: application of coin operator C followed by the application of shift operator S. The shift operator is a unitary transformation defined on $\mathcal{H}_c \otimes \mathcal{H}_v$ such that $S\,|a, v\rangle \rightarrow |a, w\rangle$ where w is the a^{th} neighbor of v. Coin operator C is a unitary transformation on $\mathcal{H}_c$. This walk is executed t times on an undirected graph $G(V,E)$.

$$\left|\psi\right\rangle_{\text{final}} = \mathbf{U}^{t} \left|\psi\right\rangle_{\text{initial}} \tag{1}$$

where $|\boldsymbol{\psi}\rangle_{\text{initial}}$ = $|\textbf{coin, position}\rangle_{\text{initial}}$ and t is the number of applications of **U**. Our aim is to identify the marked vertex $|\mathbf{v}_{\textbf{target}}\rangle \in \mathcal{H}_v$. Success of the algorithm depends on the probability of measuring position state, $|\textbf{position}\rangle \in \mathcal{H}_v$, to observe the vertex $|\mathbf{v}_{\textbf{target}}\rangle$.

4.2. Quantum Walk Search

The following are some definitions given for the better understanding of the proposed algorithm.

Definition 7. Hypercube is a regular graph of degree d with $D = 2^d$ vertices. Each vertex is labeled by a *d-bit* binary string. Two nodes x, y in a hypercube are adjacent if and only if they differ by a single bit flip, i.e., their hamming distance is one.

Our algorithm transforms a subtree into a hypercube search space to gain speed-up in the search process. For simplicity, the average branching factor b is assumed to be a power of two. So, the Hilbert space associated with our quantum walk is $\mathcal{H} = \mathcal{H}^d \otimes \mathcal{H}^D$, where $\mathcal{H}^d$ is the d dimensional Hilbert space associated with the coin operator **C** and $\mathcal{H}^D$ is the vertex space. Most frequently used coin operator for discrete quantum walk search is the Grover's operator $\mathbf{C_0}$.

Definition 8. Grover's operator is defined as $\mathbf{C_0} = 2\,|\mathbf{S^c}\rangle\langle\mathbf{S^c}| - \mathbf{I}$ on the d-dimensional Hilbert space $\mathcal{H}^d$ with $|\mathbf{i}\rangle$ as its canonical basis, where

$$\left|S^{c}\right\rangle = \frac{1}{\sqrt{d}}\sum_{i=0}^{d-1}|i\rangle \tag{2}$$

and **I** is the unit matrix of order d.

The Shift operator **S** moves the walker from its state $|\mathbf{a}\rangle|\vec{v}\,\rangle$ to $|\mathbf{a}\rangle|\vec{v} \oplus \vec{e}_{\,a}\rangle$, where $\vec{e}_{\,a}$ is a d-bit binary string with zeroes at all positions except a^{th} bit and $\oplus$ is the bit-wise **XOR** operator.

$$\hat{S}:|a\rangle|\vec{v}\rangle \rightarrow |a\rangle|\vec{v}\oplus\vec{e}_a\rangle \tag{3}$$

Another representation of Shift operator is

$$\hat{S} = \sum_{a=1}^{d}\sum_{\vec{v}=0}^{2^d-1}|a\rangle|\vec{v}\oplus\vec{e}_a\rangle\langle a,\vec{v}| \tag{4}$$

Since, the coin is symmetric in all d directions equally, a perturbation is introduced in the coin operator for marked vertices.

Definition 9. Perturbed walk $\mathbf{U'} = \mathbf{S}.\mathbf{C'}$ is a unitary transformation with oracular coin $\mathbf{C_1} = \mathbf{-I_d}$ for the marked vertex and Grover's coin $\mathbf{C_0}$ for the unmarked vertices, where

$$\mathbf{C' = C_0 \otimes (I_d - |v_{target}\rangle\langle v_{target}|) + C_1 \otimes |v_{target}\rangle\langle v_{target}|} \tag{5}$$

Inhomogeneity in the perturbed walk makes it capable of identifying $|\mathbf{v_{target}}\rangle$ at the end of algorithm.

4.3. Quantum Walk Search

The quantum walk search algorithm to compute subgame perfect equilibrium in a two-player sequential game follows.

Algorithm **Compute SGPE(*T*)**

begin
label all leaf nodes with alphabets $l_1, l_2, \ldots l_n$ and leave internal nodes unlabeled
for each farthest subtree T_i of tree T **do**
begin
index leaf nodes of T_i with d-bit binary strings // let 2^d be the smallest power of 2 greater than b^2
generate uniform superposition state:

$$\left|0\right\rangle_c \otimes \left|0\right\rangle_s \rightarrow \frac{1}{\sqrt{dD}} \sum_{i=0}^{d-1} \sum_{j=0}^{D-1} \left|i\right\rangle_c \otimes \left|j\right\rangle_s$$

perform $t_f = (\pi/2)(\sqrt{2^d})$ steps of the quantum walk **U'** = **S**. **C'** with the propagator

$$\mathbf{S} = \sum_{d,x} \left| d, \vec{x} \oplus \vec{e}_d \right\rangle \left\langle d, \vec{x} \right|$$

and the perturbed coin $\mathbf{C' = C_0 \otimes I + (C_1 - C_0) \otimes |x_{target}\rangle\langle x_{target}|}$
measure the final state in $|\mathbf{d}, \vec{x}\rangle$ basis
replace sub-tree T_i by the root of T_i updated with the label and pay-off of the measured solution
end for
end procedure

Fig.1. Psuedo-code for Compute SGPE

Algorithm shown in Fig.1 takes a game tree *T* of average branching factor *b* and height *h* as input and gives a leaf node as output which is the subgame perfect equilibrium of *T*.

5. Results and Discussion

We consider a finite two player game in its extensive form for the application of the quantum walk algorithm. It is assumed that the branches are uniformly distributed and all the leaf nodes are present at same level. The main result of applying the proposed algorithm is given below:

Theorem 1. There exists an algorithm to compute subgame perfect equilibrium in a two-player game tree of average branching factor b and height h with O(n/b) queries.

Proof. We start our proof using the result of Shenvi et.al's work [14]. They proved that $t_f = (\pi/2)(\sqrt{2^d})$ steps of the quantum walk $\hat{S} = \sum_{d,x} \left| d, \vec{x} \oplus \vec{e}^{\,d} \right\rangle \left\langle d, \vec{x} \right|$ **U'** = **S**. **C'** with the propagator and the perturbed coin $\mathbf{C' = C_0 \otimes (I_d - |v_{target}\rangle\langle v_{target}|) + C_1 \otimes |v_{target}\rangle\langle v_{target}|}$ on a hypercube of dimension *d* results in the measurement of marked state with probability *(1/2) - O(1/d)*.

Conversion of leaf nodes l_i of each subtree T_i into hypercube search space enables faster search in the subtree. By this conversion, each subtree needs $(\pi/2)(\sqrt{b^2}) = (\pi/2)(b)$ queries to get equilibrium, because a smallest subtree of a two-player game contains b^2 leaf nodes in it.

At any level, number of nodes present = $b^{level\ height}$

Our algorithm starts from the leaf level and the number of subtrees at the leaf level for a duopoly game is b^{h-2}.

Continuing this process at each level, the height gets reduced by two until it reaches the root node T (where $h = 0$).

Therefore, the total number y of smallest possible sub-trees, is $b^0 + b^2 + \ldots + b^{h-2}$, which follows a geometric progression of common ratio b^2.

Therefore, $y = (b^h - 1)/(b^2 - 1)$

Hence, total number of queries applied = no. of subtrees × no. of queries per subtree $= \left(\frac{b^h - 1}{b^2 - 1}\right)\left(\frac{\pi}{2}b\right)$

Here, $(b^h - 1) \approx n$ and $(b^2 - 1) \approx b^2$

Thus, total number of queries = $O(n/b)$.

Hence, it is proved that the proposed algorithm is capable of computing subgame perfect equilibrium in a two-player game tree with $O(n/b)$ queries.

In the case of k-player game tree, the total number of smallest possible sub-trees (y), becomes $b^0 + b^k + \ldots + b^{h-k}$, if we assume all players *take turns in a circular order*. Here, y is a geometric progression with common rati b^k.

Therefore, $y = (b^h - 1)/(b^k - 1)$.

If there is an oracular coin which is capable of identifying equilibrium for a k-player subtree, then the procedure to calculate required number of queries is similar to the one followed in the proof of above theorem.

Hence, total number of queries applied $= \left(\frac{b^h - 1}{b^k - 1}\right)\left(\frac{\pi}{2}b\right)$

Thus, total number of queries = $O(n/b^{(k-1)})$.

The problem exhibits an exponential growth in the count of leaf nodes to be queried with the growth in the height of the tree. The query complexity is $O(b^h)$ for a tree with b as branching factor and h as height. The above result shows that the performance of our algorithm is $O(b)$ times faster than the best existing algorithm for two-player games. It is convenient to proceed to the discussion after examining Table 1 that compares classical and quantum query complexities of some real games.

Table 1. Comparison of Query Complexities in Two Player Sequential Games

Name of the game	Branching factor (b)	Height (h)	O(bh) = O(n) (power of 10)	O(n/b) (power of 10)
Connect6	46000	30	140	135
Arimaa	17281	92	390	385
Amazons (10×10)	374	84	216	213
Hex (11×11)	280	40	98	95
Go (19×19)	250	150	360	357
Chess	35	80	123	121

As it turns out, the application of discrete-time quantum walk with backward induction is responsible for the speedup. The perturbed walk with Grover's coin $\mathbf{C_0}$ for the marked state accelerates the search in each subtree and backward induction procedure preserves correctness of the solution. Since the probability of getting the solution at each subtree ≈ 1/2, application of perturbed walk a constant number of times enhances the probability to nearly one [14]. Distribution of probability over the solution for a game tree with multiple

solutions is not examined here. Analyzing the effect of quantum walk in the presence of multiple solution is taken for future enhancement of the algorithm.

6. Conclusions

In this paper, perturbed discrete time quantum walk is applied on game trees to compute subgame perfect equilibrium. Since quantum random walk on hypercube can be applied to any regular graph, the leaf nodes of the game tree are transformed into a hypercube structure to compute subgame perfect equilibrium. Similar transformations can be made to any search space in order to apply this method successfully. We have shown that our algorithm computes the solution in a duopoly game with $O(n/b)$ queries and a k-player game with $O(n/b^{(k-1)})$ queries. Thus, the random walk based algorithm is $O(b)$ times better than the best known classical algorithm. The random walk used here is based on a discrete time walk on a hypercube of dimension d. Leaf nodes branching out of same internal nodes are transformed into a hypercube search space where the discrete quantum walk identifies the equilibrium node. This process is iterated backwards until the root node is reached.

Higher branching factors make classical algorithms computationally more complex due to the exponentially increasing number of nodes, resulting in combinatorial explosion. We provide a new perspective on designing quantum algorithms for such classical games. This work opens up a number of future directions of research. Some of the research challenges include identifying solutions if they exist in multiple numbers, improving probability of measuring the solution and achieving exponential speedup over the classical algorithms.

Acknowledgements

The authors wish to thank Hemalatha Thiagarajan and Michael Arock who provided insight and expertise that greatly assisted the research.

References

[1] Selten, R. Reexamination of the perfectness concept for equilibrium points in extensive games, Int. J. Game Theory, 1975; 4(1):25-55.
[2] Osborne, M.J., Rubinstein, A. A course in game theory, MIT press; 1994.
[3] Venegas-Andraca, S.E. Quantum walks: a comprehensive review, Quantum Inf. Process., 2012; 11(5):1015-1106.
[4] Zermelo, E. Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels, Proceedings of the fifth international congress of mathematicians, Cambridge UP, Cambridge, 1913; 2(2).
[5] Kuhn, H.W. Extensive games and the problem of information, Contributions to the Theory of Games, 1953; 2(28):193-216.
[6] Aumann, R.J. Backward induction and common knowledge of rationality, Games Econ. Behav., 1995; 8(1):6-19.
[7] Papadimitriou, C.H. Computational complexity, John Wiley and Sons Ltd., 2003.
[8] Fraenkel, A.S., Lichtenstein, D. Computing a perfect strategy for n × n chess requires time exponential in n, Automata, Languages and Programming, Springer Berlin Heidelberg, 1981; 278-293.
[9] Bopardikar, S. D., Borri, A., Hespanha, J. P., Prandini, M., Di Benedetto, M. D. Randomized sampling for large zero-sum games, Automatica, 2013; 49(5):1184-1194.
[10] Meyer, D. A. Quantum strategies, Phys. Rev. Lett., 1999; 82(5):1052.
[11] Eisert, J., Wilkens, M., Lewenstein, M. Quantum games and quantum strategies, Phys. Rev. Lett., 1999; 83(15):3077.
[12] Marinatto, L., Weber, T. A quantum approach to static games of complete information, Phys. Lett. A,

2000; 272(5):291-303.
[13] Iqbal, A., Toor, A.H. Backwards-induction outcome in a quantum game, Phys. Rev. A, 2002; 65(5):052328.
[14] Shenvi, N., Kempe, J., Whaley, K.B. Quantum random-walk search algorithm, Phys. Rev. A, 2003; 67(5):052307.
[15] Ambainis, A. Quantum walk algorithm for element distinctness, SIAM J. on Comput., 2007;37(1):210-239.
[16] Ambainis, A., Bačkurs, A., Nahimovs, N., Ozols, R., Rivosh, A. Search by quantum walks on two-dimensional grid without amplitude amplification, In: Theory of Quantum Computation, Communication, and Cryptography, Springer Berlin Heidelberg, 2013; 87-97.
[17] Tani, S., Claw finding algorithms using quantum walk, Theor. Comput. Sci., 2009;410(50):5285-5297.
[18] Aaronson, S., Ambainis, A. Quantum search of spatial regions, In: Proceedings. 44th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2003; 200-209.

Authors' Profiles

Arish Pitchai graduated with Bachelor of Computer Applications degree from Madurai Kamaraj University and Master of Computer Applications degree from National Institute of Technology Tiruchirappalli. Presently, he is pursuing Ph.D degree in Computer Applications in National Institute of Technology Tiruchirappalli.

A V Reddy received his M.E degree in Computer Science from National Institute of Technology Tiruchirappalli and Ph.D degree in Applied Mathematics form Indian Institute of Science, Bangalore. He is currently working as a Professor in Department of Computer Applications at National Institute of Technology Tiruchirappalli. He has successfully supervised 4 doctoral students in their research program.

Nickolas Sarvarimuthu received M.E and Ph.D degrees from National Institute of Technology Tiruchirappalli. He is currently an Associate Professor and Head of Computer Applications at the National Institute of Technology Tiruchirappalli. He is the Professor In-Charge of the Massively Parallel Programming Laboratory, NVIDIA CUDA Teaching Centre, National Institute of Technology Tiruchirappalli.

How to cite this paper: Arish Pitchai, A V Reddy, Nickolas Savarimuthu,"Quantum Walk Algorithm to Compute Subgame Perfect Equilibrium in Finite Two-player Sequential Games", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.2, No.3, pp.32-40, 2016.DOI: 10.5815/ijmsc.2016.03.03