# Online appendix: Algorithm for finite-action two-player games

This appendix describes the algorithm that computes the grid-invariant equilibrium for finite-action two-player games, and for any finite grid $g$, in a similar spirit to Appendix B of the paper. One can also use this algorithm to introduce its limiting version, that is, as the fineness of the grid $\varphi(g)$ goes to zero.[1] A Matlab code for the limiting version of this algorithm is also available online, at *http://www.stanford.edu/~leinav.*

This algorithm and the one presented in Appendix B of the paper are similar, but are not the same. Solving for an arbitrary number of actions requires significant adaptations. Most importantly, the mechanics of the two-by-two algorithm follow more closely the concepts of critical points and associated stages defined in the paper (Definitions 1 and 2): each additional step in the algorithm of Appendix B is associated with a new single stage. In contrast, this new algorithm sometimes merges multiple stages into one. Loosely speaking, this happens when critical points "trigger" each other, and therefore converge to each other as the fineness of the grid $\varphi(g)$ goes to zero.[2] Another adaptation is in the search for the next critical point (part 2 of the algorithm). With two actions, we only needed to consider a binary decision by each player: whether to switch at $(a, t\,)$ or not. With more actions, this decision also includes the choice of which action to switch into.

Let us make a few additional comments. First, for simplicity of computation and notation, we have written this algorithm (and the Matlab code) for the set of cost technologies that can be described by $C_p(a_p \to a'_p, t) = c(t)$. It would be easy to extend it for some more general families. For example, we could easily allow cost technologies that can be described by $C_p(a_p \to a'_p, t) = \theta_p^{a_p \to a'_p} c(t)$ (where the $\theta_p^{a_p \to a'_p}$'s are a set of parameters). However, more work would be needed to accommodate other types of cost technology. Second, we introduce some notation. As in Appendix B of the paper, if $p$ is one player we use $\sim p$ to denote the other player. We also define $\text{nextround}_p(t) \equiv \text{next}_p(\text{next}_{\sim p}(t))$. In words, $\text{nextround}_p(t)$ is the first point in time after $t$ at which player $p$ can play after the other player has already moved. We also use this operator recursively, so that $\text{nextround}_p^n(t)$ would mean applying this operator $n$ times.

Given a particular game $(\Pi, c, g)$ with $K_1$ and $K_2$ actions for players 1 and 2, respectively, the algorithm steps are described below.

**Initialization:** Set $m = 0$ (stage counter, starting from the end); $t_0^* = T$ (the last critical time encountered); $V_0(a, p) = \Pi$ (continuation value of player $p$ at profile $a$ just before $t_0^*$); $AM_0(a, p) = \varnothing$ (a function that indicates if there is an active switch just before time $t_m^*$ by player $p$ from profile $a$, and, if there is, to which action. With some abuse of notation, we designate this by $\varnothing$ when there is no active switch).

[1] Since the switching cost technology is continuous, the limiting version is almost identical to the finite version of the algorithm. Unless otherwise noted, the only changes are that $\text{next}_i(t)$ and $\text{prev}_i(t)$ are replaced by $t$.

[2] As an example, running this algorithm on the motivating example of Section 2, the stages described in the third and fourth row of the table in that section would be associated with a single step of the algorithm.

**Update** $(m, V_m, AM_m)$**:**

1. $m = m + 1$

2. Find the next critical time $t^*_m$, and the action $a^*$ and player $p^*$ associated with it.[3] This is done by comparing the potential benefits and costs for each move.

   (a) We use some auxiliary definitions:

   i. Let $q(a,p)$ be the first player who switches out of $a$ if player $\sim p$ is the first who moves. More precisely, let

$$q(a,p) = \begin{cases} \sim p & \text{if } AM_{m-1}(a, \sim p) \neq \varnothing \\ p & \text{if } AM_{m-1}(a, \sim p) = \varnothing \text{ and } AM_{m-1}(a,p) \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases}$$

   ii. Let $SM_{m-1}(a,p)$ be the longest ordered set of action profiles $(a^0, a^1, ..., a^{k-1}, a^k)$ such that $a^0 = a$ and, for $i > 0$

$$a^i = \begin{cases} (a^{i-1}_{\sim q(a,q)}, AM_{m-1}(a^{i-1}, q(a,q))) & \text{if } i \text{ is odd and } AM_{m-1}(a^{i-1}, q(a,q))) \neq \varnothing \\ (a^{i-1}_{q(a,q)}, AM_{m-1}(a^{i-1}, \sim q(a,q))) & \text{if } i \text{ is even and } AM_{m-1}(a^{i-1}, q(a,q)) \neq \varnothing \end{cases}$$

   This defines the sequence of consecutive switches within stage $m-1$ that starts at $a$ and ends at a profile from which there is no active move. We denote this final node by $\overline{SM}_{m-1}(a,p)$. The sequence is finite and is solely a function of $AM_{m-1}$.

   iii. Given $SM_{m-1}(a,p) = (a^0, ..., a^k)$, define $FS_{m-1}(a,p) = \sum_{i=1}^{k} I(a^{i-1}_p \neq a^i_p)$ where $I(\cdot)$ is the indicator function. $FS_{m-1}$ computes the number of switches by player $p$ in the $SM_{m-1}(a,p)$ sequence.

   iv. Let $\Delta V_{m-1}(a,p,b_p) \equiv V_{m-1}(\overline{SM}_{m-1}((b_p, a_{\sim p}), p)) - V_{m-1}(\overline{SM}_{m-1}(a,p))$. This difference in values stands for the benefit (or loss) of a switch by player $p$ from profile $a$ to action $b_p \in A_p - \{a_p\}$, without accounting for the switching cost of such a move. Let also $\Delta FS_{m-1}(a,p,b_p) \equiv FS_{m-1}((b_p, a_{\sim p}), p) - FS_{m-1}(a,p)$. This difference stands for the difference in the number of subsequent immediate moves by player $p$, when considering a move from profile $a$ to action $b_p$.

   (b) We now use these definitions to find the next critical time:

   i. First, we compute $tt_m(a,p,b_p)$, the first point in time at which player $p$ would find it profitable to switch from profile $a$ to action $b_p$. This involves three different cases, as shown below. The first is when the switch gives a negative value (and therefore is never profitable). The second is a case in which the difference in values favors a switch, and, moreover, if player $p$ does not switch, he will be making more immediate subsequent switches than if he switches. This means that player

[3] This part of the algorithm corresponds to the subroutine *FindNextStage* in the associated Matlab code.

$p$ would prefer to switch right away rather than staying put, so $tt_m(a,p,b_p)$ kicks in immediately. The third case is the "standard" case, in which the critical time is the latest point on the gird at which the cost of switching is less than its benefit.[4]

$$tt_m(a,p,b_p)^5 = \begin{cases} 0 \\ \quad \text{if } \Delta V_{m-1}(a,p,b_p) < 0 \text{ or } (\Delta V_{m-1}(a,p,b_p) = 0 \text{ and } \Delta FS_{m-1}(a,p,b_p) \geq -1) \\ \text{prev}_p(t^*_{m-1}) \\ \quad \text{if } \Delta V_{m-1}(a,p,b_p) \geq 0 \text{ and } \Delta FS_{m-1}(a,p,b_p) < 0\text{, except for} \\ \quad (\Delta V_{m-1}(a,p,b_p) = 0 \text{ and } \Delta FS_{m-1}(a,p,b_p) = -1) \\ \max\left\{t \in g_p, t < t^*_{m-1} | c(t) + \sum_{i=FS_{m-1}(a,p)+1}^{FS_{m-1}((b_p,a_{\sim p}),p)} c(\text{nextround}^i_p(t)) \leq \Delta V_{m-1}(a,p,b_p)\right\} \\ \quad \text{if } \Delta V_{m-1}(a,p,b_p) > 0 \text{ and } \Delta FS_{m-1}(a,p,b_p) \geq 0 \end{cases}$$

ii. Next, we compute for each profile $a$ what is the latest time at which it is profitable to switch out from this profile by each player. We make sure not to account for those switches which were already active. More precisely, for each $(a,p)$ we compute

$$t_m(a,p) = \begin{cases} 0 & \text{if } AM_{m-1}(a,p) = \underset{b_p \in A_p \backslash \{a_p\}}{\arg\max}\ tt_m(a,p,b_p) \\ \underset{b_p \in A_p \backslash \{a_p\}}{\max} tt_m(a,p,b_p) & \text{otherwise} \end{cases}$$

iii. Finally, we compute the next critical time by finding the latest such time across profiles and players. That is, we compute:

$(a^*,p^*) = \underset{(a,p)}{\arg\max}\{t_m(a,p)\}$, and

$t^* = \underset{(a,p)}{\max} t_m(a^*,p^*)$

(c) Given $(a^*,p^*)$:

**Terminate** if $t^* = 0$, and if so set also $\overline{m} = m-1$.

**Abort** if $|p^*| > 1$.[6] This implies that there are equal critical times for different players, and that the solution is not grid invariant.

Otherwise, set $t^*_m = t^*$ and $p^*_m = p^*$, and continue to part 3.

[4] When $\Delta FS_{m-1}(a,p,b_p) > 0$ one has to account for multiple switches. It is in this last case when the assumption that $c(t)$ is common to all players and moves helps considerably. To accommodate richer families of cost technologies, one would need to introduce cumbersome notation to keep track of the costs associated with switching along the $SM$ sequence, instead of simply counting them.

[5] Note that by having weak inequalities within the max operator we implicitly assume that a player switches whenever he is indifferent between switching or not.

[6] arg max is a correspondence. Given the way we construct $t_m(a,p)$, the multiple solutions must be associated with a unique $p^*$ for any finite grid. In the limiting case, this is the only generic case. This is why the algorithm may abort in non-generic cases.

3. We now enter a ("short") phase in which the set of active switches gets computed at every point in the grid until it "stabilizes," as defined below. This is where (potentially) multiple stages of the game (as defined in Definition 2) are computed within a single step of the algorithm. This part of the algorithm is considerably different from the one described in Appendix B of the paper.

   (a) First, let

   $V^{temp}(a,p) = V_{m-1}(\overline{SM}_{m-1}(a, \sim p^*_m)) - \sum_{i=0}^{FS_{m-1}(a,\sim p^*_m)} c(\text{nextround}^i_p(t^*_m))$

   These are the continuation values for each player and profile just after $t^*_m$.

   (b) Second, use these continuation values and apply standard backward induction on the game grid from time $t^*_m$ backwards. At each point in time, record the full set of equilibrium strategies. Stop applying backward induction when the strategies for both players remain constant for a full round. More precisely, stop at the latest $t \in g_p$ that satisfies the following conditions: (i) $\text{next}(t) \in g_{\sim p}$; (ii) the strategies for $p$ at $t$ and $\text{next}_p(t)$ are the same; (iii) the strategies for $\sim p$ at $\text{next}(t)$ and $\text{nextround}_{\sim p}(\text{next}(t))$ are also the same; and (iv) $\text{nextround}_{\sim p}(\text{next}(t)) \leq t^*_m$.[7]

   (c) Finally, update $AM_m$ and $V_m$. The strategies for player $p$ at $t$ and those of $\sim p$ at $\text{next}(t)$ constitute $AM_m$. The continuation values for player $p$ at $\text{next}(p)$ and those of $\sim p$ at $t$ constitute $V_m$.

   **Terminate** if $\sum_{(a,p)} I\left(AM_m(a,p) \neq \varnothing\right) = K_1(K_2-1) + K_2(K_1-1)$ where $I(\cdot)$ is the indicator function. This condition implies that the maximal possible switches are active. Before termination, let also $\overline{m} = m$, $t^*_{\overline{m}+1} = 0$. Otherwise, go to part 1.

**Output:** The essential information of the algorithm consists of the number of stages of the game, $\overline{m}$, the critical points that define the end of each stage, $(t^*_m)_{m=0}^{\overline{m}}$, the strategies at every stage $(AM_m)_{m=0}^{\overline{m}}$, and the continuation values at the earliest stage, $V_{\overline{m}}$. The initial equilibrium actions are given by the (generically) unique profile $a_{initial}$ that has no active switches at the earliest stage, i.e. $AM_{\overline{m}}(a_{initial}, p) = \varnothing$ for $p = 1, 2$. The equilibrium path can be computed by tracing the sequence along the $SM_m(a,p)$'s, stage by stage. In other words, start at $a_{initial}$, go to $\overline{SM}_1(a_{initial}, p)$, then continue to $\overline{SM}_2(\overline{SM}_1(a_{initial}, p), p)$, and so on. The values for the game are given by $V_{\overline{m}}(a_{initial}, p)$ for each player $p$.

[7] In the case of the limiting algorithm this step is done by letting the players take alternate moves, all of them taking place at time $t^*_m$. This is correct since we know that when a player has consecutive moves, his incentives do not change.