Optimizing the Video Game Multi-Jump: Player Strategy, AI, and Level Design

Aaron M. Broussard, Martin E. Malandro, and Abagayle Serreyn

Abstract. This article initiates the mathematical study of multi-jumping in video games. We begin by proving a necessary, and frequently sufficient, condition for a multi-jump to be *optimal*, i.e., achieve the highest possible height after traveling a given horizontal distance. We then give strategies that can be used by human players and by AI to select successful multi-jumps in real time. We also show how a video game designer can build the ground around a platform to guarantee that the platform is reachable—or unreachable—by a multi-jump beginning at any point on the ground.

1. INTRODUCTION. In many platform-based video games the player is able to perform a *double jump*, which is a normal jump followed by a second jump initiated in midair without the aid of a platform. The arc of the second jump might be identical to that of the first, it might be a smaller version of the first and otherwise be subject to the same gravitational pull, or it might have its own arc and obey a completely different gravitational law. For example, Capcom®'s *Devil May Cry*™[1] features the first kind of double jump, Konami®'s *Castlevania®: Symphony of the Night*™ features the second, and Klei Entertainment®'s *Mark of the Ninja*™ features the third. In this paper we study multi-jumps, which generalize double jumps. A *multi-jump* is a finite sequence of jumps where the first jump is initiated from the ground and the rest are initiated in midair. The number of jumps in a multi-jump is the *length* of the multi-jump, so a double jump is a multi-jump of length two. Several video games, such as Chair Entertainment Group®'s *Shadow Complex*™ and Nintendo®'s *Super Smash Bros.*™ *Melee*, feature triple jumps or multi-jumps of even longer length.

The basic problem we consider in this paper is the following. Suppose that a character in a two-dimensional side-scrolling video game wishes to use a multi-jump to jump to the right from a fixed starting point across a gap and land on a fixed platform. By a *platform* we always mean an impenetrable horizontal platform (so the character cannot pass through the bottom of the platform) that begins at a point and extends indefinitely to the right. In most games the character can control the horizontal velocity component of her jumps. We assume that the target platform begins far enough to the right from the character that she will utilize the maximum horizontal velocity possible for each jump. We therefore assume that the character has a known finite sequence of jump arcs available to her and faces the problem of selecting when to jump in midair, i.e., to switch from the arc of one jump to the next, so as to land on the platform. See Figures 1–2, in which the character can jump twice in midair.

Provided the platform is reachable by a multi-jump, we give strategies for solving this problem on the fly for both player-controlled and artificial intelligence (AI)-controlled characters. In the simplest situation all jumps available to the character are equal and fully concave (Definition 5). In this situation we give a simple strategy (the *line method*) that is usable by both players and AI. In our experience the majority of

http://dx.doi.org/10.4169/amer.math.monthly.123.10.1013
MSC: Primary 90C27, Secondary 91A55; 97A20

[1] All products, company names, brand names, and trademarks are properties of their respective owners.

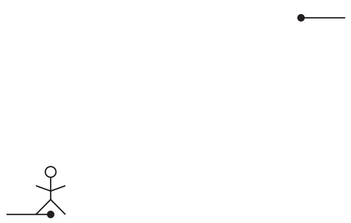


Figure 1. Character wishes to jump to distant platform

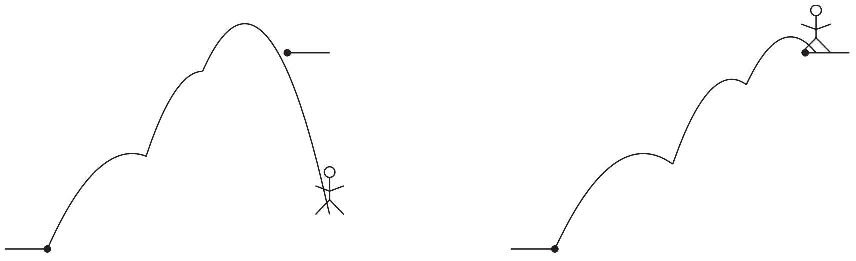

(A) An unsuccessful multi-jump (B) A successful multi-jump

Figure 2. Multi-jumps

games featuring multi-jumps are covered by this situation. We give two further strategies for AI-controlled characters in more-complicated situations. Our first AI strategy is very general, in that it applies to *any* collection of standard jump functions (Definition 1). We also give a faster (less computationally intensive) AI strategy for collections of standard jump functions whose derivative inverses are known and computable exactly.

We have evidence that our AI results are new, or at least previously unknown to game developers: We have observed that the multi-jumping AI in *Super Smash Bros.*™ *Melee* is not optimal, in that there are situations in the game where the AI will consistently select a multi-jump that fails to cross a gap even though such a multi-jump is possible. The more recent games in Nintendo®'s *Super Smash Bros.*™ series (*Super Smash Bros.*™ *Brawl* and *Super Smash Bros.*™ *for Wii U*™) feature better, but still not optimal, multi-jumping AI. Due to the online nature of *Super Smash Bros.*™ *for Wii U*™ it is possible that the AI in this game could be improved in a future update. While multi-jumping is a common feature in video games, the only games we could find that feature real-time multi-jumping AI are the games in the *Super Smash Bros.*™ series.

In reality, platforms have finite length. We use the assumption of infinite-length platforms only to justify the correctness of our player and AI strategies, and if our strategies would cause a character to overshoot a finite-length platform, the same strategies could be applied to land successfully on such a platform by either lowering the horizontal velocity of the character or by initiating the multi-jump earlier, i.e., farther to the left.

We also consider applications to game level design. Given a fixed platform and a fixed sequence of jumps available to the player, we consider the problem of how to design the ground around the platform so that the platform is reachable—or unreachable—by a multi-jump starting at any point on the ground. Platform-based adventure games where the player gains new abilities as she explores the map and uses these new powers to reach previously inaccessible areas are frequently referred to in the gaming community as *Metroidvanias* [**9**]. (This term is a combination of *Metroid*™ and *Castlevania*®, which are two famous series of games featuring similar gameplay.) In a typical Metroidvania the player eventually gains the ability to jump a second or third time in midair. Our results can be used, for example, to place a platform just tantalizingly out of reach of a player who is only able to jump once (or twice), but which is easily reached when the player gains the ability to jump a second (or third) time.

We note that while we carry out our analysis in two dimensions, our results are directly applicable to multi-jumps occurring in a two-dimensional plane in any three-dimensional game.

The study of the complexity of games and the development of AI for playing games have rich histories. For instance, computers have been playing games of strategy against humans for over 35 years, and have become sophisticated enough to challenge the world's best players. IBM®'s Deep Blue® famously bested former World Chess Champion Garry Kasparov in a 1997 series [**4**]. More recently, Google® DeepMind™'s AlphaGo AI [**6**] beat Lee Sedol, the world's top Go player over the previous decade, four to one in a five-game series [**2**]. There is also a wealth of research directly applicable to video game AI. For instance, pathfinding algorithms such as the A* algorithm [**3**] are important for AI in real-time strategy and first-person shooter games. As for complexity, a number of video games, including generalized versions of *Metroid*™, have been shown to be NP-hard—see [**1**] and the references therein. To our knowledge this paper marks the first time that the problem of optimal multi-jumping in video games has been studied.

2. JUMP FUNCTIONS AND MULTI-JUMPS.

Jump functions are the building blocks of multi-jumps. The graph of a jump function captures the trajectory that a character (as represented by a point in space) would follow by starting at the origin and jumping to the right.

Definition 1. A *jump function* is a continuous function $f : \mathbb{R}_{\geq 0} :\rightarrow \mathbb{R}$ for which $f(0) = 0$ and $\exists c > 0$ such that:

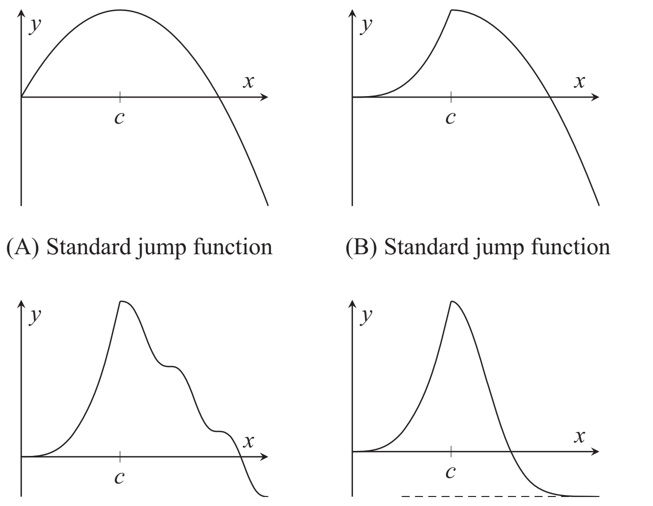
- f is strictly increasing on $[0, c]$, and
- f is continuously differentiable on $[c, \infty)$, with $f'(x) \leq 0$ for $x \in (c, \infty)$ (so f is weakly decreasing on $[c, \infty)$) and $f'_+(c) = 0$.

We say that f *peaks at* c. Additionally if f is concave down (i.e., f' is strictly decreasing) on $[c, \infty)$ and $\lim_{x\to\infty} f'(x) = -\infty$, then we say f is a *standard* jump function.

This terminology is our own, as jumps in video games have not been studied formally before. Note that if f is a standard jump function, then f is automatically strictly decreasing on $[c, \infty)$ and $\lim_{x\to\infty} f(x) = -\infty$. By definition, if f is a jump function we have $f'_+(c) = 0$. For convenience we will write $f'(c) = 0$, which will make our central result (Theorem 2) easier to state.

Examples of graphs of both standard and nonstandard jump functions may be found in Figure 3. It is easy to generate jump functions. For instance, let $a, c, k > 0$, $r > 1$, and let $g(x)$ be any strictly increasing continuous function for which $g(0) = 0$ and $g(c) = k$. Then

$$f(x) = \begin{cases} g(x) & \text{if } 0 \leq x < c, \\ -a(x-c)^r + k & \text{if } x \geq c \end{cases} \tag{1}$$



(A) Standard jump function (B) Standard jump function

(C) Nonstandard jump function (D) Nonstandard jump function

Figure 3. Examples of jump function graphs

is a standard jump function that peaks at c. Jump functions of the form (1) will appear again in Sections 4, 5, and 6.

Unless otherwise stated, without loss of generality we assume that the character initiates her multi-jump at the origin $(0, 0)$. Fix a sequence $F = (f_1, \dots, f_n)$ of jump functions.

Definition. If $(x_1, \dots, x_{n-1})$ is a sequence of nonnegative real numbers, then the piecewise function $F_{(x_1,\dots,x_{n-1})} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ given by

$$F_{(x_1,\dots,x_{n-1})}(x) = \begin{cases} f_1(x) & \text{if } 0 \leq x \leq x_1, \\ f_2(x - x_1) + f_1(x_1) & \text{if } x_1 \leq x \leq x_1 + x_2, \\ f_3(x - x_1 - x_2) + f_2(x_2) + f_1(x_1) & \text{if } x_1 + x_2 \leq x \leq x_1 + x_2 + x_3, \\ & \vdots \\ f_{n-1}(x - \sum_{i=1}^{n-2} x_i) + \sum_{i=1}^{n-2} f_i(x_i) & \text{if } \sum_{i=1}^{n-2} x_i \leq x \leq \sum_{i=1}^{n-1} x_i, \\ f_n(x - \sum_{i=1}^{n-1} x_i) + \sum_{i=1}^{n-1} f_i(x_i) & \text{if } \sum_{i=1}^{n-1} x_i \leq x \end{cases}$$

is called the *multi-jump defined by* $(x_1, \dots, x_{n-1})$.

The graph of $F_{(x_1,\dots,x_{n-1})}$ captures the trajectory a character would follow by starting at the origin and jumping to the right n times, following the arcs of f_1 through f_n in sequence, where the arc of the first jump is followed for x_1 horizontal units, the arc of the second is followed for x_2 horizontal units, and so on, and the arc of f_n is followed indefinitely.

Given $x_n \geq 0$, we denote by $F_{(x_1,\dots,x_n)}$ the restriction of $F_{(x_1,\dots,x_{n-1})}$ to the domain $[0, \sum_{i=1}^{n} x_i]$. That is, the graph of $F_{(x_1,\dots,x_n)}$ is the same as the graph of $F_{(x_1,\dots,x_{n-1})}$, except that in $F_{(x_1,\dots,x_n)}$ the final arc of the multi-jump is followed for only x_n units. The $x_1, \dots, x_n$ are called *jump points*, and $(x_1 + \cdots + x_n, F_{(x_1,\dots,x_n)}(x_1 + \cdots + x_n))$ is called the *ending point* of the multi-jump. Note that the final jump point x_n does *not* initiate a new jump, and that the ending point of a multi-jump might be in midair.

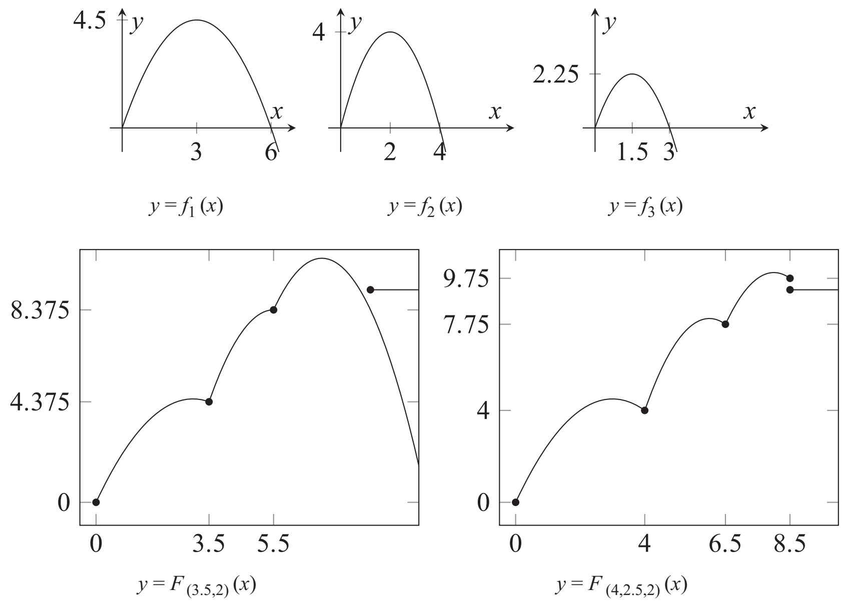


Figure 4. Graphs of $F_{(x_1,x_2)}$ and $F_{(x_1,x_2,x_3)}$ for the jump functions in the Main Example, together with the platform beginning at (8.5, 9.25)

Also note that $F_{(x_1,\ldots,x_n)}(x_1 + \cdots + x_n) = f(x_1) + \cdots f(x_n)$, and that the maximum possible height achievable by a multi-jump is $f_1(c_1) + \cdots + f_n(c_n)$, achievable by jumping at peaks—that is, by selecting the jump points $(c_1, \ldots, c_n)$. In summary, given a sequence of n jump functions, a sequence of $n - 1$ jump points defines a multi-jump with no ending point, while a sequence of n jump points defines a multi-jump with an ending point. In general n can be any fixed integer $n \geq 1$. In this paper we will draw all of our figures with $n = 3$.

Main Example, Part 1. For the Main Example let us take the following sequence of jump functions. Let

$$f_1(x) = -0.5(x - 3)^2 + 4.5,$$

$$f_2(x) = -(x - 2)^2 + 4,$$

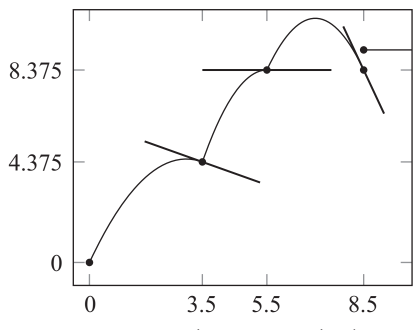
$$f_3(x) = -(x - 1.5)^2 + 2.25.$$

These functions are the building blocks for Figure 2, in which the multi-jumps begin at (0, 0) and the distant platform begins at (8.5, 9.25). The graph in Figure 2(a) is the graph of $F_{(3.5,2)}$, while the graph in Figure 2(b) is the graph of $F_{(4,2.5,2)}$, with domain extended slightly to the right so that the final arc touches the platform. See Figure 4.

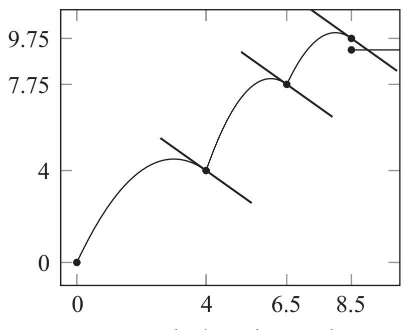
We note that there is no requirement for the f_i to be quadratics (or even polynomials) in general. We have chosen them to all be quadratics for this example simply for ease of hand calculations later in the paper.

3. THE FUNDAMENTAL THEOREM.

In this section we state and prove what we call the fundamental theorem of multi-jumping optimization. This theorem applies to all jump functions and has far-reaching consequences, as it and its primary consequence (Theorem 3) form the basis for the design of all of our player and AI



$F_{(3.5,2,3)}$ is a non-optimal multi-jump ending at $x = 8.5$.



$F_{(4,2.5,2)}$ is the unique optimal multi-jump ending at $x = 8.5$; hmax $(8.5) = 9.75$.

Figure 5. Optimal and nonoptimal multi-jumps, drawn together with the platform beginning at (8.5, 9.25) and tangent lines at jump points

strategies and are integral ingredients in our proofs throughout the rest of the paper. Let $(f_1, \ldots, f_n)$ be a sequence of jump functions, where f_i peaks at $c_i \in \mathbb{R}_{>0}$. Let $C = c_1 + \cdots + c_n$. Recall our assumption that multi-jumps begin at the origin (0, 0).

Theorem 2 (The fundamental theorem of multi-jumping optimization). *Let $d \in \mathbb{R}$ such that $d \geq C$. Then there exists a largest value $\text{hmax}(d) \in \mathbb{R}$ such that the platform beginning at $(d, \text{hmax}(d))$ is reachable by a multi-jump, and if $(x_1, \ldots, x_n)$ is any sequence of jump points for which $F_{(x_1,\ldots,x_n)}$ ends at $(d, \text{hmax}(d))$, then $x_i \geq c_i$ for $i \in \{1, \ldots, n\}$, and*

$$f_1'(x_1) = f_2'(x_2) = \cdots = f_n'(x_n).$$

Remark. In summary, the fundamental theorem says that for a multi-jump to be *optimal*—that is, achieve the maximum possible height after traveling a specified horizontal distance d—the slope of the tangent line to the character's trajectory at the ending point of the multi-jump must equal the slope of the tangent line to her trajectory at every jump point along the way. See Figure 5, which revisits the multi-jumps from Figure 4. (In Figure 5, that $F_{(4,2.5,2)}$ is in fact optimal and is the unique optimal multi-jump ending at $x = 8.5$ will be shown in the Main Example, Part 2 at the end of this section.)

Optimal multi-jumps corresponding to a sequence of increasing d values are shown in Figure 6. This figure demonstrates the interplay between d and $\text{hmax}(d)$, in that $\text{hmax}(d)$ is a (generally weakly) decreasing function of d. This figure also shows that $\text{hmax}(d)$ can be negative when d is large enough. From a gameplay perspective $\text{hmax}(d)$ being negative makes perfect sense—sometimes the target platform is at a lower height than the character's starting position. Also note from this figure that when $\text{hmax}(d) = 0$, the jump points for an optimal multi-jump do not have to occur along the line $y = 0$.

Proof of Theorem 2. First we show the existence of $\text{hmax}(d)$ using a bit of topology. Let $X = [0, d]^n \subset \mathbb{R}^n$ and let $Y = \{(x_1, \ldots, x_n) \in X : x_1 + \cdots + x_n = d\}$.

Y is an $n - 1$-simplex, so Y is compact. Since the function $H : [0, \infty)^n \to \mathbb{R}$ given by $H(x_1, \ldots, x_n) = f_1(x_1) + \cdots + f_n(x_n)$ is continuous, and the continuous image of

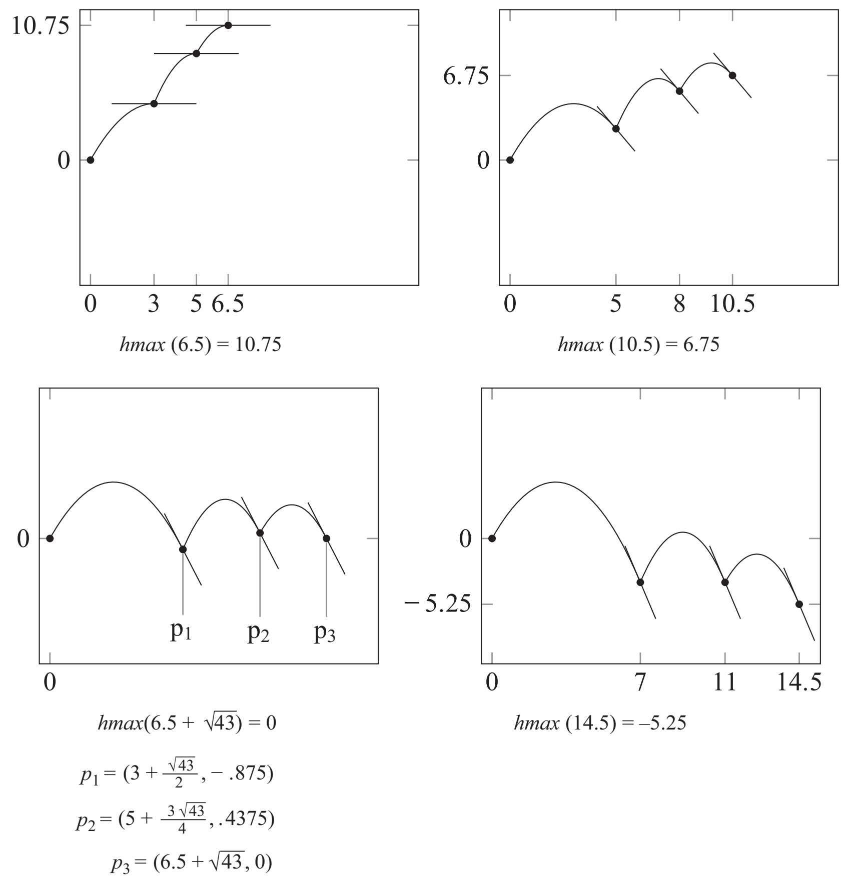


Figure 6. Optimal multi-jumps corresponding to selected d-values; jump functions from the Main Example

a compact set is compact, H attains a maximum value on Y. This maximum value is by definition $\mathrm{hmax}(d)$.

Let $(x_1, \dots, x_n)$ be a sequence of jump points such that $F_{(x_1,\dots,x_n)}$ ends at $(d, \mathrm{hmax}(d))$, i.e., for which $x_1 + \cdots + x_n = d$ and $f_1(x_1) + \cdots + f_n(x_n) = \mathrm{hmax}(d)$.

Next we show that $x_i \geq c_i$ for all i. Suppose not, so for some i we have $0 \leq x_i < c_i$. Since $x_1 + \cdots + x_n = d \geq C = c_1 + \cdots + c_n$, for some j we have $x_j > c_j$. Let $\epsilon > 0$ such that $x_i + \epsilon < c_i$ and $x_j - \epsilon > c_j$. Then the sequence of jump points $(\overline{x_1}, \dots, \overline{x_n})$ given by

$$\overline{x_k} = \begin{cases} x_i + \epsilon & \text{if } k = i, \\ x_j - \epsilon & \text{if } k = j, \\ x_k & \text{otherwise} \end{cases}$$

has the property that $\overline{x_1} + \cdots + \overline{x_n} = d$. Furthermore, since $f_i(\overline{x_i}) > f_i(x_i)$ and $f_j(\overline{x_j}) \geq f_j(x_j)$ we have $f_1(\overline{x_1}) + \cdots + f_n(\overline{x_n}) > f_1(x_1) + \cdots + f_n(x_n) = \mathrm{hmax}(d)$, contradicting the maximality of $\mathrm{hmax}(d)$. Therefore $x_i \geq c_i$ for all i.

Finally we show that $f_1'(x_1) = f_2'(x_2) = \cdots = f_n'(x_n)$.

If $x_i > c_i$ for all i we may apply the method of Lagrange multipliers in $\mathbb{R}^n$ (see, e.g., [**7**, Ex. 5-16]) to the domain $(c_1, \infty) \times \cdots \times (c_n, \infty) \subset \mathbb{R}^n$ of the objective function $H(x_1, \dots, x_n) = f_1(x_1) + \cdots + f_n(x_n)$, subject to the constraint $g(x_1, \dots, x_n) =$

0, where $g(x_1, \ldots, x_n) = x_1 + \cdots + x_n - d$. Applying Lagrange multipliers, we immediately obtain the system of equations

$$\begin{aligned} f_1'(x_1) &= \lambda, \\ f_2'(x_2) &= \lambda, \\ &\vdots \\ f_n'(x_n) &= \lambda \end{aligned}$$

for some $\lambda \in \mathbb{R}$, so $\lambda = f_1'(x_1) = f_2'(x_2) = \cdots = f_n'(x_n)$.

On the other hand, if $x_i = c_i$ for some i, then $f_i'(x_i) = 0$. Suppose for the sake of contradiction that $f_j'(x_j) \neq 0$ for some j. Then $x_j > c_j$ and $f_j'(x_j) < 0$. For the sake of clarity we give the idea of the rest of the argument, which can be made rigorous in a straightforward, albeit somewhat lengthy manner. Since f_i' and f_j' are continuous on $[c_i, \infty)$ and $[c_j, \infty)$, $f_i'(x_i) = 0$, and $f_j'(x_j) < 0$, it is possible to bump x_i slightly to the right and x_j slightly to the left to arrive at a contradiction. In particular, due to the continuity of the derivatives it is possible to choose points $\overline{x_i}$ and $\overline{x_j}$ such that $\overline{x_i} > x_i$, $c_j < \overline{x_j} < x_j$, $\overline{x_i} + \overline{x_j} = x_i + x_j$, and $f_i(\overline{x_i}) + f_j(\overline{x_j}) > f_i(x_i) + f_j(x_j)$. It then immediately follows that for the sequence of jump points given by $(\overline{x_1}, \ldots, \overline{x_n})$ (where $\overline{x_k} = x_k$ for $k \neq i, j$), we have $\overline{x_1} + \cdots + \overline{x_n} = d$ and $f_1(\overline{x_1}) + \cdots + f_n(\overline{x_n}) > f_1(x_1) + \cdots + f_n(x_n) = \text{hmax}(d)$, contradicting the maximality of $\text{hmax}(d)$. It follows that $0 = f_1'(x_1) = f_2'(x_2) = \cdots = f_n'(x_n)$, completing the proof. ∎

The rest of the paper will focus on standard jump functions. While the fundamental theorem provides a necessary condition for a multi-jump to be optimal, for standard jump functions it is also sufficient, as the following very useful consequence demonstrates.

Theorem 3. *Suppose $f_1, \ldots, f_n$ are standard. Write $f_i'^{-1}$ for the inverse of the restriction of f_i' to $[c_i, \infty)$. Let $d \in \mathbb{R}$ such that $d \geq C$. Then for some unique $x_1 \geq c_1$ we have*

$$d = x_1 + \sum_{i=2}^{n} f_i'^{-1}(f_1'(x_1)),$$

$$\text{hmax}(d) = f_1(x_1) + \sum_{i=2}^{n} f_i(f_i'^{-1}(f_1'(x_1))),$$

and the unique multi-jump ending at $(d, \text{hmax}(d))$ is defined by the sequence of jump points

$$(x_1, f_2'^{-1}(f_1'(x_1)), \ldots, f_n'^{-1}(f_1'(x_1)))$$

(i.e., the sequence $(x_1, x_2, \ldots, x_n)$ for which $f_1'(x_1) = f_2'(x_2) = \cdots = f_n'(x_n)$).

Proof. Since f_i is standard, f_i' restricted to $[c_i, \infty)$ is invertible, and as a function of $x_1 \in [c_1, \infty)$, the function $x_1 + \sum_{i=2}^{n} f_i'^{-1}(f_1'(x_1))$ is increasing, continuous, and has range $[C, \infty)$. Hence there exists unique $x_1 \geq c_1$ such that $d = x_1 + \sum_{i=2}^{n} f_i'^{-1}(f_1'(x_1))$.

Suppose the sequence of jump points $(z_1, \dots, z_n)$ defines a multi-jump ending at $(d, \mathrm{hmax}(d))$. Then $d = \sum_{i=1}^n z_i$ and $\mathrm{hmax}(d) = \sum_{i=1}^n f_i(z_i)$. By the fundamental theorem $z_i \geq c_i$ for all i, and for $i \geq 2$ we have $f_1'(z_1) = f_i'(z_i)$. Hence $z_i = f_i'^{-1}(f_1'(z_1))$. Since $\sum_{i=1}^n z_i = d$, we have $z_1 = x_1$, so

$$(z_1, \dots, z_n) = (x_1, f_2'^{-1}(f_1'(x_1)), \dots, f_n'^{-1}(f_1'(x_1)))$$

is the unique sequence of jump points defining a multi-jump ending at $(d, \mathrm{hmax}(d))$. Furthermore, we have $\mathrm{hmax}(d) = \sum_{i=1}^n f_i(z_1) = f_1(x_1) + \sum_{i=2}^n f_i(f_i'^{-1}(f_1'(x_1)))$. ■

Remark 4. For standard jump functions, Theorem 3 reduces the problem of finding optimal multi-jumps to the problem of analyzing the function $x_1 + \sum_{i=2}^n f_i'^{-1}(f_1'(x_1))$ of the single variable x_1.

Main Example, Part 2. The functions in the Main Example are all standard. Let us use Theorem 3 to find hmax(8.5) as well as the jump points x_1, x_2, x_3 such that the multi-jump $F_{(x_1,x_2,x_3)}$ ends at (8.5, hmax(8.5)). We compute

$$\begin{aligned} &f_1'(x) = -x + 3, && \\ &f_2'(x) = -2x + 4, && f_2'^{-1}(x) = -\frac{x}{2} + 2, \\ &f_3'(x) = -2x + 3, && f_3'^{-1}(x) = -\frac{x}{2} + \frac{3}{2}. \end{aligned}$$

We seek x_1 such that $x_1 + f_2'^{-1}(f_1'(x_1)) + f_3'^{-1}(f_1'(x_1)) = 8.5$, that is,

$$x_1 + \left(-\frac{-x_1 + 3}{2} + 2\right) + \left(-\frac{-x_1 + 3}{2} + \frac{3}{2}\right) = 8.5,$$

which yields $x_1 = 4$. We then obtain $x_2 = f_2'^{-1}(f_1'(4)) = 2.5$ and $x_3 = f_3'^{-1}(f_1'(4)) = 2$. Finally we get

$$\mathrm{hmax}(8.5) = f_1(4) + f_2(2.5) + f_3(2) = 9.75,$$

so $F_{(4,2.5,2)}$ is the unique multi-jump ending at $(8.5, \mathrm{hmax}(8.5)) = (8.5, 9.75)$, as shown in Figure 5. The jump points for the graphs in Figure 6 were computed in an analogous fashion.

4. PLAYER STRATEGY.

Definition 5. A jump function f is *fully concave* if f is standard and f is concave down on $[0, \infty)$.

Of the jump functions appearing in Figure 3, only (A) is fully concave. Let $(f_1, \dots, f_n)$ be a sequence of jump functions, where f_i peaks at c_i. Let $C = c_1 + \cdots + c_n$. Fix $d \geq C$ and $h \in \mathbb{R}$. Consider the problem of selecting a multi-jump to reach a platform beginning at (d, h), assuming the platform is reachable. In this section we give a simple and complete solution to this problem, implementable in real time by human players (and AI), in the case that $f_1 = \cdots = f_n$ are fully concave jump functions. This is the case for a majority of games that feature multi-jumps, probably

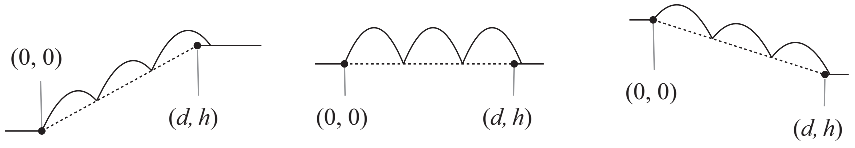


Figure 7. The line method selects successful multi-jumps.

because this scenario is the easiest to implement by a game programmer. In Section 5 we will address this problem for AI-controlled characters in the relaxed situation that $f_1, \dots, f_n$ are standard and not necessarily equal.

Suppose f_1 is a fully concave jump function. Write $f = f_1$, $c = c_1$, and consider the sequence of jump functions $(\underbrace{f, \dots, f}_{n})$. Here is our strategy, which we call the *line method*. Consider the line connecting $(0, 0)$ and (d, h). Whenever the character's trajectory intersects this line, jump. See Figure 7.

Theorem 6. *If the platform beginning at (d, h) is reachable, then the line method will select a multi-jump that causes the character to land on the platform.*

Proof. By Theorem 3 the unique sequence of jump points $(z_1, \dots, z_n)$ defining the multi-jump ending at $(d, \text{hmax}(d))$ has $z_1 = \cdots = z_n$. Let $z = z_1$, so $nz = d$. Since $d \geq C = nc$ we have $z \geq c$. Since f is concave down it lies above any of its secant lines, so the following "strategy" selects the jump points $(z, \dots, z)$: Consider the line $\hat{L}$ connecting $(0, 0)$ and $(d, \text{hmax}(d))$. Whenever the character's trajectory intersects $\hat{L}$, jump. (This is not a strategy a player could follow because $\text{hmax}(d)$ is not known to the player.)

Since the platform beginning at (d, h) is reachable we have $h \leq \text{hmax}(d)$. The slope of $\hat{L}$ is $\frac{\text{hmax}(d)}{d}$, while the slope of the line L for the line method is $\frac{h}{d}$. Since $h \leq \text{hmax}(d)$ and $d > 0$ we have $\frac{h}{d} \leq \frac{\text{hmax}(d)}{d}$. In particular since $\hat{L}$ intersects the downward (and not the upward) trajectory of the character's first jump, L also intersects the downward (and not the upward) trajectory of the character's first jump. Induction establishes that the line method selects jump points $(a, \dots, a)$ with $a \geq z$.

Let $F = F_{(\underbrace{a, \dots, a}_{n})}$ and consider the graph $y = F(x)$ of the multi-jump selected by the line method. Using again the fact that the graph of f lies above any of its secant lines and that $na \geq nz = d$ we have that the graph $y = F(x)$ from $x = 0$ to $x = d$ is always above (or touching) L. Therefore, d horizontal units into this trajectory the character will be at the point $(d, F(d))$, with $F(d) \leq \text{hmax}(d)$ (by definition of $\text{hmax}(d)$) and $F(d) \geq h$ (since L intersects (d, h)). That is, d horizontal units into the trajectory selected by the line method the character will either be above or touching the platform. Therefore, since the platform extends indefinitely to the right and $\lim_{x \to \infty} f(x) = -\infty$, continuing to follow this trajectory to the right will eventually cause the character to land on the platform. ∎

Remark. We require the assumption that f is concave down throughout its domain to ensure that the trajectory selected by the line method will not collide with the underside of the platform. For example, if

$$f(x) = \begin{cases} x^3 & \text{if } x \leq 1, \\ -(x-1)^2 + 1 & \text{if } x > 1, \end{cases}$$

then for the sequence of jump functions (f, f) we have $C = 2$, and the platform beginning at $(2, 1)$ is easily reachable by a double jump (by jumping at the peak, for instance). However one may check that jumping according to the line method—that is, jumping in midair when the line connecting $(0, 0)$ and $(2, 1)$ intersects the downward trajectory of the first jump—would cause the character to collide with the underside of the platform.

Remark. Suppose the platform beginning at (d, h) is reachable. When the jump functions $f_1, \dots, f_n$ available are not fully concave or not all equal, we have been unable to give a strategy that a human player could follow that *guarantees* that the player will reach the platform. However in this situation the fundamental theorem still provides a useful heuristic—try to imagine a multi-jump that ends at or above (d, h), and jump whenever the slope of the tangent to your current jump matches the slope of the tangent of the ending point of the imagined multi-jump.

5. AI STRATEGY.

Let $(f_1, \dots, f_n)$ be a sequence of standard jump functions, where f_i peaks at c_i. Let $C = c_1 + \cdots + c_n$. Fix $d \geq C$ and $h \in \mathbb{R}$. Consider the problem of deciding whether the platform beginning at (d, h) is reachable by a multi-jump, and if it is, deciding what multi-jump to use to reach the platform.

We now give a numerical solution to this problem which is efficient enough to be used in real time by an AI-controlled character. Following Remark 4, our strategy is to analyze the function $g(x_1) = x_1 + \sum_{i=2}^{n} f_i'^{-1}(f_1'(x_1))$ to find the sequence of jump points $(x_1, \dots, x_n)$ defining the multi-jump that ends at $(d, \text{hmax}(d))$. Then the platform beginning at (d, h) is reachable if and only if $h \leq \text{hmax}(d)$, and if the platform is reachable, then continuing to follow the multi-jump ending at $(d, \text{hmax}(d))$ to the right will eventually land the character on the platform (since the platform extends indefinitely to the right and $\lim_{x\to\infty} f_n(x) = -\infty$).

Algorithm 1 implements this approach. It accepts $\epsilon > 0$, $f_1, \dots, f_n$, $c_1, \dots, c_n$, and $d \geq c_1 + \cdots + c_n$, and computes $(x_1, \dots, x_n)$ such that, with error less than ϵ, the multi-jump defined by $(x_1, \dots, x_n)$ ends at $(d, \text{hmax}(d))$. Specifically, the output of Algorithm 1 is guaranteed to satisfy $|d - \sum_{i=1}^{n} x_i| < \epsilon$ and $|f_i'(x_i) - f_1'(x_1)| < \epsilon$ for all i.

We assume we have access to a function `root`$(f(x), a, b, e)$, which accepts a continuous function $f(x)$ on $[a, b]$ and returns $z \in [a, b]$ such that $f(r) = 0$ for some $r \in [a, b]$ with $r \in [z - e, z + e]$, and throws an exception if no such $z \in [a, b]$ exists. Such a function is available in any number of scientific computational packages—see, e.g., `scipy` [**5**]. We also assume that the f_i and f_i' can be evaluated exactly.

Algorithm 1. Algorithm for computing the jump points $(x_1, \dots, x_n)$ defining the multi-jump ending at $(d, \text{hmax}(d))$.

```
#Input: $(f_1, \dots, f_n), (c_1, \dots, c_n), d, \epsilon$
#Output: $(x_1, \dots, x_n)$

def NumericalInverse($f, y, a, b, e$):
        $g(x) = f(x) - y$
        while True:
                try:
                        return root($g(x), a, b, e$)
```

```
 9 except RuntimeError (no root): #if there is no root on $[a,b]$
10 $b = b \times 10$ #then expand the search range to the right
11
12 def OptimalJumpPoints($(f_1,\dots,f_n), (c_1,\dots,c_n), d, \epsilon$):
13         $e = \epsilon$
14         def DerivativeCheck($(x_1,\dots,x_n)$):
15                 for $i = 2,\dots,n$:
16                         if $|f_i'(x_i) - f_1'(x_1)| \geq \epsilon$:
17                                 return False
18                 return True
19         $F(x) = x - d + \sum_{i=2}^{n}$ NumericalInverse($f_i', f_1'(x), c_i, d, e/n$)
20         while True:
21                 $x_1 =$ root($F(x), c_1, d, e/n$)
22                 for $i = 2,\dots,n$:
23                         $x_i =$ NumericalInverse($f_i', f_1'(x_1), c_i, d, e/n$)
24                 if $|d - \sum_{i=1}^{n} x_i| < \epsilon$ and DerivativeCheck($(x_1,\dots,x_n)$):
25                         return True
26                 else:
27                         $e = e \times 0.1$
28
29 return OptimalJumpPoints($(f_1,\dots,f_n), (c_1,\dots,c_n), d, \epsilon$)
```

In Algorithm 1 we assume that the $f_i'^{-1}$ must be evaluated numerically. If they can be evaluated exactly, a handful of simple changes makes the algorithm significantly more efficient:

Algorithm 2. Modification of Algorithm 1 when the $f_i'^{-1}$ can be computed exactly.

- The functions `NumericalInverse` and `DerivativeCheck` are no longer necessary.
- `NumericalInverse`$(f_i', f_1'(x), c_i, d, e/n)$ is replaced with $f_i'^{-1}(f_1'(x))$ on line 19.
- `NumericalInverse`$(f_i', f_1'(x_1), c_i, d, e/n)$ is replaced with $f_i'^{-1}(f_1'(x_1))$ on line 23.
- The phrase "and `DerivativeCheck`$((x_1, \dots, x_n))$" is removed from line 24.

We have implemented Algorithms 1 and 2 in `Sage`, a free and open-source computer algebra system [**8**], and have found our implementations to be quite fast in practice across a wide range of standard jump functions.

For purposes of comparison we tested Algorithms 1 and 2 with functions of the form given by (1). Each of our two tests consisted of 2500 samples, where a sample consisted of a sequence of jump functions $(f_1, f_2, , \dots, f_n)$ (with each f_i of the form $f_i(x) = -a_i(x - c_i)^{r_i} + k_i$), d a random real number in $\left[\sum_{i=1}^{n} c_i, 1.5\sum_{i=1}^{n} c_i\right]$, and a set value of ϵ. We ran each sample 50 times in each of our algorithms and took the time for the 40th-slowest run. That is, 80% of calls to our algorithms ran at least as quickly as the numbers reported here. We chose to report at the 80th percentile level (as opposed to the slowest-seen level) to give reasonable running time estimates—the slowest calls are sometimes slowed by system-specific scheduling issues that have nothing to do with the intrinsic speed of our algorithms. We ran our tests in a single thread on an Intel® Core™ i5-6600 processor at stock speed.

To establish a baseline for comparison, our first test was $n = 3$, with $a_i \in [0.5, 10]$, $c_i \in [1, 10]$, $k_i \in \{1, \dots, 10\}$ (each selected uniformly at random), $r_i = 2$, and $\epsilon = 10^{-9}$.

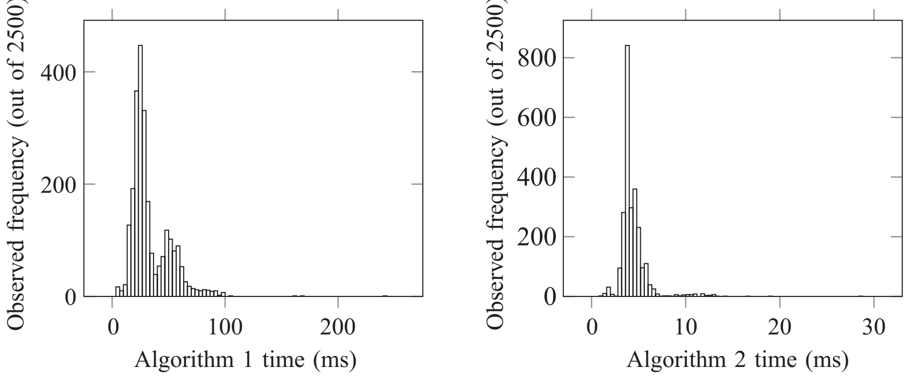


Figure 8. Algorithm test 2

In this test both algorithms were extremely fast, with Algorithm 1 always completing in less than 10.1 milliseconds and Algorithm 2 in less than 1.7 milliseconds.

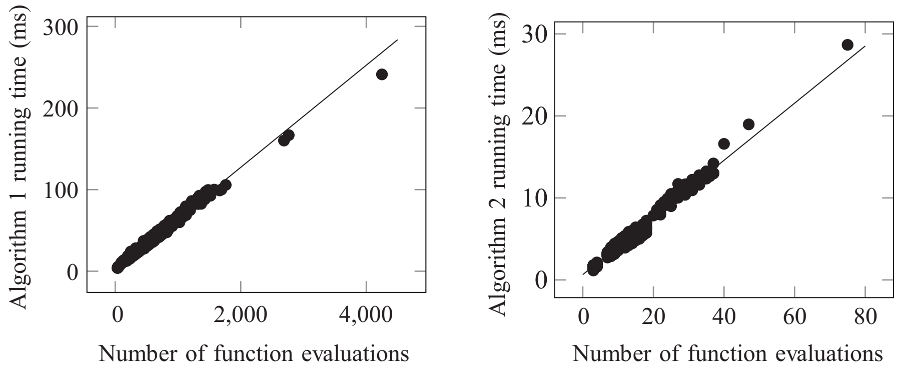


Figure 9. Algorithm test 2

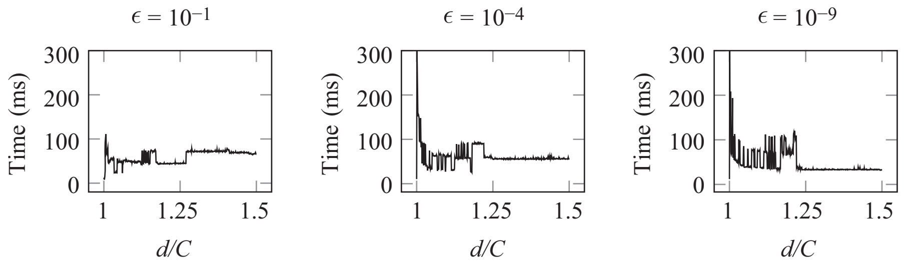


Figure 10. Algorithm 1 speeds for the Main Outlier: $n = 4$ with $f_1(x) = -9.76(x - 5.81)^2 + 7$, $f_2(x) = -8.17(x - 7.83)^5 + 5$, $f_3(x) = -8.44(x - 7.59)^5 + 1$, $f_4(x) = -6.16(x - 7.64)^4 + 3$

In our second test we went beyond parabolas, taking $n = 4$ and selecting $r_i \in \{2, 3, 4, 5\}$ uniformly at random. We continued to take $a_i \in [0.5, 10]$, $c_i \in [1, 10]$, $k_i \in [1, 10]$ (each selected uniformly at random), and $\epsilon = 10^{-9}$. In this test Algorithm 1 always completed within 242 milliseconds, while Algorithm 2 always

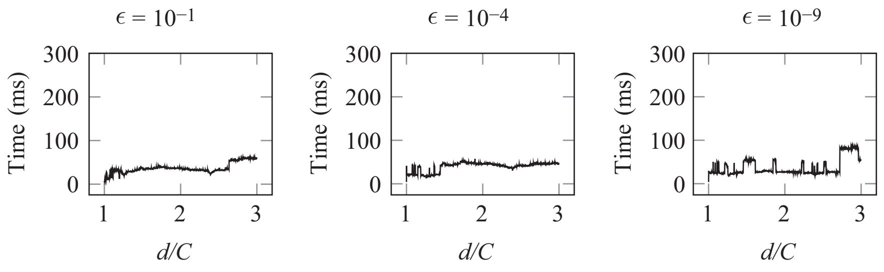


Figure 11. Algorithm 1 speeds for $n = 4$ for a typical sample: $f_1(x) = -5.05(x - 8.66)^4 + 8$, $f_2(x) = -5.02(x - 3.30)^3 + 4$, $f_3(x) = -5.95(x - 3.75)^2 + 8$, $f_4(x) = -5.35(x - 2.75)^4 + 6$

completed within 29 milliseconds. Median completion times were 27 milliseconds and 4 milliseconds, respectively. See Figure 8.

The primary factor affecting the running times of our algorithms is the number of function evaluations made during root finding. Scatter plots of the number of function evaluations versus overall algorithm speed in our second test are given in Figure 9, together with their lines of best fit. For each algorithm we see three outliers (which are in fact the same three samples), the worst of which required over 4000 function evaluations in Algorithm 1. That sample was, to two decimal places (which is enough to replicate this behavior),

$$f_1(x) = -9.76(x - 5.81)^2 + 7, \qquad f_2(x) = -8.17(x - 7.83)^5 + 5,$$
$$f_3(x) = -8.44(x - 7.59)^5 + 1, \qquad f_4(x) = -6.16(x - 7.64)^4 + 3.$$

We will call this the Main Outlier. For each of our outliers we found that d was only very slightly larger than C, and that either increasing d slightly or decreasing ϵ was enough to make our algorithms run with times comparable to the other samples in our tests. In Figure 10 we show the running time of Algorithm 1 for the Main Outlier across a range of d and ϵ values. The type of information in Figure 10 is the most pertinent type of information for a game's AI developer, who works with a fixed collection of jump functions. This information can vary considerably depending on the specific collection of jump functions. For instance, in Figure 11 we display the running times of a typical sample in our tests across a range of d and ϵ values. If that sample were the collection of jump functions in our video game, we could be confident that Algorithm 1 would complete quickly given any reasonable combination of d and ϵ input values.

6. LEVEL DESIGN.

In this section we consider multi-jumps beginning at points other than the origin. Let $(f_1, \ldots, f_n)$ be a sequence of jump functions and $(x_1, \ldots, x_n)$ a sequence of nonnegative real numbers. The *multi-jump beginning at* (a, b) *defined by* $(x_1, \ldots, x_n)$, denoted $F^{(a,b)}_{(x_1,\ldots,x_n)}$, is the translation of $F_{(x_1,\ldots,x_n)}$ horizontally by $|a|$ units (left if $a < 0$ and right if $a > 0$) and vertically by $|b|$ units (up if $b > 0$ and down if $b < 0$). The graph of $F^{(a,b)}_{(x_1,\ldots,x_n)}$ captures the trajectory a character would follow by starting at (a, b) and jumping to the right n times, following the arcs of f_1 through f_n in sequence, where the arc of the first jump is followed for x_1 horizontal units, the arc of the second is followed for x_2 horizontal units, and so on, and the arc of f_n is followed for x_n units. The multi-jump $F^{(a,b)}_{(x_1,\ldots,x_n)}$ covers a total horizontal distance of $x_1 + \cdots + x_n$ units and ends at a height of $b + f(x_1) + \cdots + f_n(x_n)$.

Suppose now that $f_1, \dots, f_n$ are standard, $x_i \geq c_i$ for all i, and $f_1'(x_1) = f_2'(x_2) = \cdots = f_n'(x_n)$. Let $d = \sum_{i=1}^n x_i$ and $h = \mathrm{hmax}(d)$. By Theorem 3, of all multi-jumps beginning at $(0, 0)$, the multi-jump defined by $(x_1, \dots, x_n)$ is the unique multi-jump ending at (d, h). In this section we address the question: How can we design the ground G so that for any point $(\overline{x}, G(\overline{x}))$ on the ground, the unique optimal multi-jump starting at $(\overline{x}, G(\overline{x}))$ and ending at $x = d$ also has the property that it ends at $y = h$? Answering this question allows a game designer to place a platform which is guaranteed to be reachable from any point on the ground (by having the platform begin at or just below (d, h)), or more interestingly, to place a platform which is guaranteed to be just barely unreachable from every point on the ground (by having the platform begin just above (d, h)).

Let $C = c_1 + \cdots + c_n$. Suppose that $\overline{x} \in \mathbb{R}$ with $d - \overline{x} \geq C$, and that the unique optimal multi-jump beginning at $(\overline{x}, \overline{y})$ and ending at (d, h) is defined by jump points $(\overline{x_1}, \dots, \overline{x_n})$. We have $f_1'(\overline{x_1}) = f_2'(\overline{x_2}) = \cdots = f_n'(\overline{x_n})$,

$$\overline{x_1} + \sum_{i=2}^{n} f_i'^{-1}(f_1'(\overline{x_1})) = d - \overline{x}, \tag{2}$$

and

$$f_1(\overline{x_1}) + \sum_{i=2}^{n} f_i(f_i'^{-1}(f_1'(\overline{x_1}))) = h - \overline{y}. \tag{3}$$

By (2) $\overline{x_1}$ is determined by $\overline{x}$, so by (3) $\overline{y}$ is determined by $\overline{x}$. That is, there is only one possible design for the ground having the required property. Write $\overline{y} = G(\overline{x})$. One can deduce $G(\overline{x})$ by solving (2) for $\overline{x_1}$ and then using (3) to find $\overline{y} = G(\overline{x})$. For most collections of jump functions this will require a numerical solution, but when the f_i are all of the form given by (1) and have the same degree, for instance, an exact solution can be given.

Theorem 7. *Suppose $f_i(x) = -a_i(x - c_i)^r + k_i$ for $x \geq c_i$ for all i, with $a_i, c_i, k_i > 0$ and $r > 1$. Then*

$$G(\overline{x}) = \left[\sum_{i=1}^{n} a_i^{1/(1-r)}\right]^{1-r} (d - C - \overline{x})^r + \left[h - \sum_{i=1}^{n} k_i\right]$$

defines the unique ground shape $\{(\overline{x}, G(\overline{x})) : d - \overline{x} \geq C\}$ for which the optimal multi-jump beginning at $(\overline{x}, G(\overline{x}))$ and ending at $x = d$ also ends at $y = h$.

Before we prove Theorem 7 we give several remarks.

Remark. In Theorem 7 the shape of the ground is a polynomial of the same degree as each of the f_i. In the case that $r = 2$, the shape of the ground is a parabola and the formula

$$G(\overline{x}) = \left[\frac{1}{\sum_{i=1}^{n} \frac{1}{a_i}}\right] (d - C - \overline{x})^2 + \left[h - \sum_{i=1}^{n} k_i\right]$$

is given in vertex form.

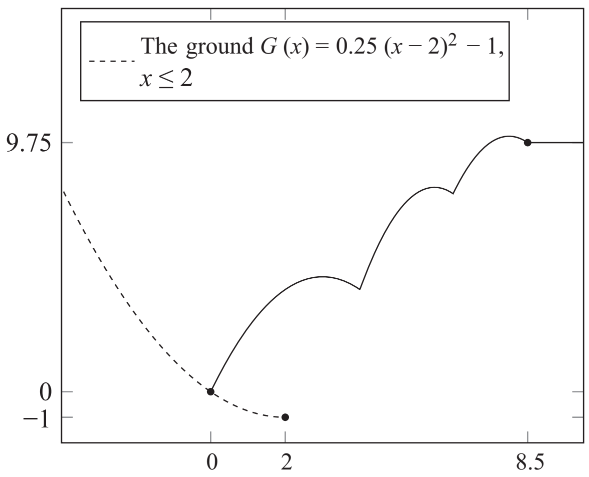


Figure 12. The optimal multi-jump beginning at (0, 0) and ending at (8.5, *hmax*(8.5) = 9.75), drawn together with the ground *G* and the platform beginning at (8.5, 9.75); jump functions from the Main Example

Main Example, Part 3. Using the functions in the Main Example with $d = 8.5$ and $h = \text{hmax}(8.5) = 9.75$, we obtain

$$\begin{aligned} G(\overline{x}) &= \left[\frac{1}{\frac{1}{1/2}+\frac{1}{1}+\frac{1}{1}}\right](8.5-(3+2+1.5)-\overline{x})^2+[9.75-(4.5+4+2.25)] \\ &= \frac{1}{4}(2-\overline{x})^2-1, \end{aligned}$$

with $\overline{x} \le 2$. In Figure 6 we graph G together with $F_{(4,2.5,2)}$, the unique optimal multi-jump beginning at (0, 0) and ending at (8.5, 9.75). In Figure 6 we add to this the graphs of several other optimal multi-jumps beginning along the ground G, all of which end at $x = 8.5$ and hence at (8.5, 9.75).

Remark. The reader may have noticed that in Figure 6, the slope of the tangent line to the ground at the starting point of any multi-jump appears to be equal to the slope of the tangent line at every jump point on that multi-jump. Indeed they are equal, and this is no coincidence—in general, as long as the ground G is continuously differentiable with $G'(x) \le 0$ for $x \le d - C$, one may regard a character walking along G as following the decreasing portion of the first jump of a multi-jump and apply the fundamental theorem to conclude that if $F^{(x,G(x))}_{(\overline{x_1},\ldots,\overline{x_n})}$ is an optimal multi-jump that begins at $(x, G(x))$ and ends at (d, h), then $G'(x) = f_1'(\overline{x_1}) = \cdots = f_n'(\overline{x_n})$.

Remark. By definition, any ground that lies strictly below G is a ground shape from which the platform beginning at (d, h) is unreachable using any multi-jump. See Figure 14.

Proof of Theorem 7. Let G be the shape of the ground having the desired property. Since $f_i'(x) = -ra_i(x-c_i)^{r-1}$ we have

$$f_i'^{-1}(x) = \left(\frac{-x}{ra_i}\right)^{1/(r-1)} + c_i,$$

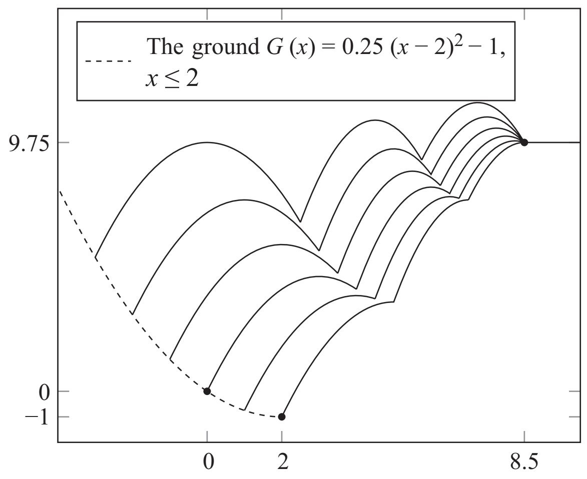


Figure 13. All optimal multi-jumps beginning along G and ending at $x = 8.5$ end at $(8.5, \text{hmax}(8.5))$; jump functions from the Main Example

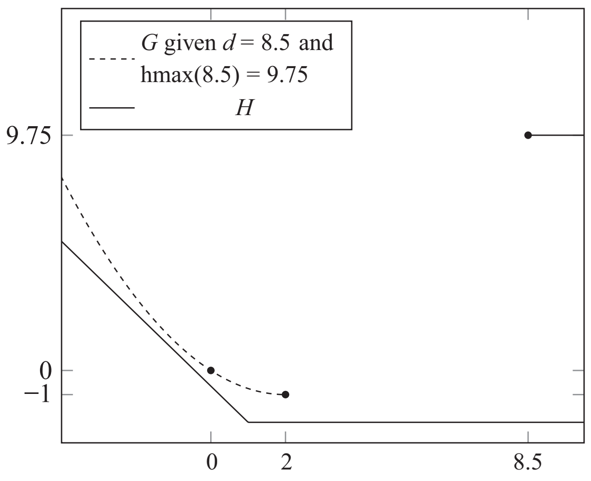


Figure 14. A ground design H from which no multi-jump can reach the platform beginning at (8.5, 9.75)

and hence

$$f_i'^{-1}(f_1'(\overline{x_1})) = \left(\frac{a_1}{a_i}\right)^{1/(r-1)} (\overline{x_1} - c_1) + c_i. \tag{4}$$

By (2), then, we have

$$\overline{x_1} + \sum_{i=2}^{n} \left(\left(\frac{a_1}{a_i}\right)^{1/(r-1)} (\overline{x_1} - c_1) + c_i \right) = d - \overline{x}.$$

Solving for $\overline{x_1}$ we obtain

$$\begin{aligned}
\overline{x_1} &= \frac{d-\overline{x}+c_1\sum_{i=2}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}-\sum_{i=2}^n c_i}{1+\sum_{i=2}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}}\\
&= \frac{d-\overline{x}+c_1\sum_{i=1}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}-\sum_{i=1}^n c_i}{\sum_{i=1}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}}\\
&= \frac{d-C-\overline{x}+c_1\sum_{i=1}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}}{\sum_{i=1}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}}\\
&= \frac{d-C-\overline{x}}{\sum_{i=1}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}}+c_1.
\end{aligned}$$

That is,

$$\overline{x_1}-c_1=\frac{d-C-\overline{x}}{\sum_{i=1}^n\left(\frac{a_1}{a_i}\right)^{1/(r-1)}}. \tag{5}$$

It follows from (4) that $f_i(f_i'^{-1}(f_1'(\overline{x_1})))=\frac{-a_1^{r/(r-1)}}{a_i{}^{1/(r-1)}}(\overline{x_1}-c_1)^r+k_i$, so by (5) we have

$$\begin{aligned}
f_i(f_i'^{-1}(f_1'(\overline{x_1}))) &= \frac{-a_1^{r/(r-1)}}{a_i{}^{1/(r-1)}\left(\sum_{j=1}^n\left(\frac{a_1}{a_j}\right)^{1/(r-1)}\right)^r}(d-C-\overline{x})^r+k_i\\
&= \frac{-1}{a_i{}^{1/(r-1)}\left(\sum_{j=1}^n\left(\frac{1}{a_j}\right)^{1/(r-1)}\right)^r}(d-C-\overline{x})^r+k_i\\
&= \frac{-a_i{}^{1/(1-r)}}{\left(\sum_{j=1}^n a_j{}^{1/(1-r)}\right)^r}(d-C-\overline{x})^r+k_i.
\end{aligned}$$

Putting this together with (3), we have

$$\begin{aligned}
G(\overline{x})=\overline{y}=h-\sum_{i=1}^n f_i(f_i'^{-1}(f_1'(\overline{x_1})))\\
&= h-\sum_{i=1}^n\left(\frac{-a_i{}^{1/(1-r)}}{\left(\sum_{j=1}^n a_j{}^{1/(1-r)}\right)^r}(d-C-\overline{x})^r+k_i\right)\\
&= \left[\sum_{i=1}^n\frac{a_i{}^{1/(1-r)}}{\left(\sum_{j=1}^n a_j{}^{1/(1-r)}\right)^r}\right](d-C-\overline{x})^r+\left[h-\sum_{i=1}^n k_i\right]
\end{aligned}$$

$$= \left[\frac{\sum_{i=1}^{n} {a_i}^{1/(1-r)}}{\left(\sum_{j=1}^{n} {a_j}^{1/(1-r)}\right)^r} \right] (d - C - \overline{x})^r + \left[h - \sum_{i=1}^{n} k_i \right]$$

$$= \left[\sum_{i=1}^{n} {a_i}^{1/(1-r)} \right]^{1-r} (d - C - \overline{x})^r + \left[h - \sum_{i=1}^{n} k_i \right]$$

as claimed. ■

7. SUMMARY AND CONCLUSIONS. We have introduced jump functions, standard jump functions, and fully concave jump functions, and have studied how to combine them optimally into multi-jumps. We have also given strategies for reaching a distant platform with a multi-jump that can be followed by any player or AI (if the multi-jump is formed from a sequence of equal fully concave jump functions), or by an AI after some quick numerical computation (if the multi-jump is formed from a sequence of standard jump functions). We have also shown how a game designer can build the ground around a platform so that the platform is reachable (or unreachable) from any point on the ground by a multi-jump.

With the exception of the fundamental theorem, our results only apply to multi-jumps formed by standard jump functions. When the jumps available to a character are nonstandard it is possible for there to be several multi-jumps ending at $(d, \text{hmax}(d))$, and the condition $f_1'(x_1) = \cdots = f_n'(x_n)$ is only a necessary condition for the sequence of jump points $(x_1, \ldots, x_n)$ to define an optimal multi-jump. This situation should be explored further. It would be interesting to give an AI strategy for selecting multi-jumps ending at $(d, \text{hmax}(d))$ when the jump functions are nonstandard. It would also be interesting to give a faster AI strategy than Algorithm 1 for standard jump functions when their derivative inverses cannot be evaluated exactly. We are currently working on an AI strategy that will apply when the jump functions are nonstandard, which we hope will also be faster than Algorithm 1.

ACKNOWLEDGMENT. The authors are grateful to the College of Sciences at Sam Houston State University for the undergraduate research award that partially supported this work and to the anonymous referees for their helpful comments.

REFERENCES

1. G. Aloupis, E. D. Demaine, A. Guo, G. Viglietta, Classic Nintendo games are (computationally) hard, *Theoret. Comput. Sci.* **586** (2015) 135–160, http://dx.doi.org/10.1016/j.tcs.2015.02.037.
2. AlphaGo: Google DeepMind, http://deepmind.com/alpha-go.html. Accessed March 23, 2016.
3. P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.* **4** (1968) 100–107, http://dx.doi.org/10.1109/TSSC.1968.300136.
4. IBM 100—Deep Blue, http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/. Accessed March 23, 2016.
5. E. Jones, T. Oliphant, P. Peterson et al., SciPy: Open source scientific tools for Python (2001–), http://www.scipy.org/. Accessed Dec. 3, 2015.
6. D. Silver et al., Mastering the game of Go with deep neural networks and tree search, *Nature* **529** (2016) 484–489, http://dx.doi.org/10.1038/nature16961.
7. M. Spivak, *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus*. Westview Press, Boulder, CO, 1965 and 1998.
8. W. A. Stein et al., Sage Mathematics Software (Version 5.6.0), The Sage Development Team (2013), http://www.sagemath.org. Accessed Dec. 3, 2015.

9. What is Metroidvania? (2015) http://www.youtube.com/watch?v=LfEOEqnYiM4. Accessed March 23, 2016.

AARON M. BROUSSARD received his B.S. in mathematics from Sam Houston State University in 2013. He then became a Simulation Software Engineer at Lockheed Martin Aeronautics and Space Systems. His free time usually involves working on puzzles with his wife, reading xkcd, math, programming, and video games.
abroussard11@gmail.com

MARTIN E. MALANDRO received his B.S. in mathematics from Texas Tech University in 2003 and his Ph.D. in mathematics from Dartmouth College in 2008. He then joined the faculty at Sam Houston State University, where he is now Associate Professor of Mathematics. In his spare time he enjoys programming, music, and the occasional video game.
Department of Mathematics and Statistics, Box 2206, Sam Houston State University, Huntsville, TX 77341-2206
malandro@shsu.edu

ABAGAYLE SERREYN is an undergraduate chemistry major and math minor at College of the Ozarks. She plans to graduate in May of 2018 and become certified to teach high school. When she is not busy doing lab work she relaxes by reading speculative fiction, solving math and logic puzzles, and singing along with her favorite songs.
abagayleserreyn@gmail.com

100 Years Ago This Month in The American Mathematical Monthly
Edited by Vadim Ponomarenko

Recent papers read before the ASSOCIATION and the SOCIETY indicate that renewed interest is apparent in all phases of mathematical history. Hence, no apology is needed for the publication of notes such as the following:

In *Nature*, December 3, 1914, p. 363, Professor CAJORI showed that the cross $\times$ as a symbol of multiplication, which is said in histories to occur first in William Oughtred's *Clavis mathematicae* (1631), is given in form of the letter x and X in Edward Wright's translation of John Napier's *Mirifici logarithmorum canonis descriptio*, second edition, London, 1618, where we read, page 4: "The note of addition is (+), of subtracting (−), of multiplying (×)." This is taken from a part of the book headed "Appendix to the Logarithmes," the authorship of which is not given but is believed now most probably to be attributed to William Oughtred.

In 1902 Professor W. W. BEMAN pointed out (*L'Intermédiaire des mathématiciens*, T. 9, Paris, p. 229, question 2424) that the colon (:) occurs as the symbol for geometric ratio at the end of the tables in Oughtred's *Trigonometria* of 1657. Professor CAJORI has found that the colon was so used by the astronomer Vincent Wing in 1651, 1655, and 1656 and by a Suffolk schoolmaster with the initials "R. B." in 1655.

The first designation of the sides of a triangle by the same letters, respectively, as the angles opposite, one group of letters being capitals A, B, C, and the other group small letters a, b, c, has been attributed to Leonhard Euler (*Histoire de l'académie de Berlin, année*, 1753, p. 231), but Professor CAJORI finds that it occurs in a pamphlet containing trigonometric formulas published by Richard Rawlinson of Queen's College, Oxford, sometime between 1655 and 1668.

—Excerpted from "Notes and News" **23** (1916) 399–404.