A gambling game arising in the analysis of adaptive randomized rounding

Richard M. Karp and Claire Kenyon

[1] UC Berkeley and ICSI
[2] Ecole Polytechnique and IUF

Abstract. Let y be a positive real number and let $\{X_i\}$ be an infinite sequence of Bernoulli random variables with the following property: in every realization of the random variables, $\sum_{i=1}^{\infty} E[X_i|X_1, X_2, \cdots, X_{i-1}] \leq y$. We specify a function $F(x, y)$ such that, for every positive integer x and every positive real y, $P(\sum_{i=1}^{\infty} X_i \geq x) \leq F(x, y)$; moreover, for every x and y, $F(x, y)$ is the best possible upper bound. We give an interpretation of this stochastic process as a gambling game, characterize optimal play in this game, and explain how our results can be applied to the analysis of multi-stage randomized rounding algorithms, giving stronger results than can be obtained using the traditional Hoeffding bounds and martingale tail inequalities.

1 Introduction

Consider the following gambling game. A player starts with a *fortune* of y and a *goal* of x. At each step the player chooses a *bet* $p \in (0, 1]$ and tosses a coin with probability of heads p. His fortune is reduced by p, and he scores a *success* if the coin comes up heads. He wins the game if he achieves x successes while maintaining a nonnegative fortune. A function $G(x, y)$ can serve as the function F mentioned in the Abstract if and only if, for all (x, y), $G(x, y)$ is an upper bound on the success probability of all strategies with fortune y and goal x. Our main result is a uniformly optimum choice of this function.

Theorem 1. *Let x be any positive integer and y, any positive real number. Let $F(x, y)$ denote the supremum, over all strategies, of the probability of achieving x successes with fortune y. Then $F(x, y)$ is specified recursively as follows.*

$$\begin{cases} \text{if } x \leq y & \text{then } F(x,y) = 1 \\ \text{if } y < x < y+1 & \text{then } F(x,y) = (x-y) + (1-x+y)F(x, x-1) \\ \text{if } y+1 \leq x & \text{then } F(x,y) = \int_{z=0}^{y} exp(-z)F(x-1, y-z)dz \end{cases}$$

Although we do not have a closed form for $F(x, y)$, we can easily compute an upper bound which is good enough for our purposes.

Corollary 1. *Let $s = (x - \lceil y \rceil)/y$. If $s > 1$, then $F(x, y) \leq (e^{s-1}/s^s)^y$.*

Proof. $F(x, y)$ is less than or equal to the probability that a Poisson random variable with mean y is greater than or equal to $x - \lceil y \rceil$. The result follows using a Chernoff bound for the tail of the Poisson distribution. □

Assume that infinitesimal bets are allowed – the precise meaning of an infinitesimal bet will be specified in Section 2. (If all bets must be positive reals then a success probability arbitrarily close to $F(x, y)$ can be achieved by placing suitably small positive bets instead of the infinitesimal bets in the strategy below.)

Theorem 2 (Best Strategy). *The following strategy achieves the success probability $F(x, y)$:*

Strategy $\mathcal{G}$ on (x, y):
if $y \geq x$ then bet 1;
if $x - 1 < y < x$ then bet $y - x + 1$;
if $y \leq x - 1$ then continue placing infinitesimal bets until a success occurs

In order to explain the link between the gambling game and multistage randomized rounding algorithms we first present an abstract setting for the traditional single-stage randomized rounding algorithms [5]. Consider a mixed integer program of the following form:

Minimize z subject to:

Integrality Constraints: $x_i \in \{0, 1\}$, $i = 1, 2, \cdots, n$;
Covering Constraints: $\sum_{i \in S_i} x_i = 1$, $i = 1, 2, \cdots, t$;
Resource Constraints: $\sum_{j \in T_j} x_j \leq z c_j$, $j = 1, 2, \cdots, m$.

Each set S_i or T_j is a subset of $\{1, 2, \cdots, n\}$, and the sets S_i are disjoint.

Each x_i represents an activity, such as the selection of a path in a graph. Each covering constraint requires that one activity be selected from a specified set; for example, in an integer multicommodity flow problem we might require that a given source-sink pair be joined by a path. Each resource constraint represents a bound on some resource; in a multicommodity flow problem the resource might be a vertex or edge, with c_j representing its nominal capacity and T_j, the set of paths that consume a unit of that capacity. The variable z represents the maximum amount by which the capacity of any resource is exceeded.

Randomized rounding begins by solving a linear programming relaxation in which the integrality constraint on each variable x_i is replaced by the constraint $0 \leq x_i \leq 1$. Let $(y_1, y_2, \cdots, y_n)$ be the optimal solution to this linear program and let z^* be the optimal value. Randomization is then used to select exactly one variable in each set S_i to be set equal to 1. Variable x_r is selected with probability y_r. This rounding process gives a feasible solution to the integer program.

Let us consider the effect of this rounding process on the resource constraints. For the jth resource constraint let $p_{ij} = \sum_{r \in S_i \cap T_j} y_r$. Then p_{ij} is the probability that a unit of resource j is used to satisfy covering constraint i. Thus the total consumption of resource j is distributed as $\sum_i X_{ij}$, where the X_{ij} are independent Bernoulli random variables and $P(X_{ij} = 1) = p_{ij}$. The expected value of the sum of these random variables is at most $z^* c_j$. The Hoeffding bound on sums of independent Bernoulli random variables is used to bound the probability that the total usage of resource j exceeds a target value zc_j and a union bound is used to obtain an upper bound on the probability that some resource exceeds its target value.

In a *multistage randomized rounding algorithm*, a sequence of mixed integer programs of the above form is solved. The resources and their capacities are the same in all these programs, but in all other respects the structure of the kth program may depend on the solutions constructed for the $k-1$ integer programs preceding it. We wish to bound the total usage of each resource over all the stages. For each resource j this total usage is a sum of Bernoulli random variables; for each set $S_i \cap T_j$ in the kth integer program there is a Bernoulli random variable with mean p_{ij}^k. Because of the adaptiveness in the choice of integer programs the Hoeffding bound, which requires that the Bernoulli random variables be independent, can only be used separately for each integer program, but not for the entire multistage process. However our gambling game, with the fortune defined as c_j times the sum of the optimal values of the linear programs, and the bets defined as the p_{ij}^k occurring in all the stages (for fixed resource j), is applicable to the multistage process because it allows the parameters of successive Bernoulli random variables to be dependent. In effect, the gambling game assumes that an adversary chooses the successive p_{ij} adaptively, with the goal of maximizing total resource usage, subject to a constraint on the sum of the p_{ij}^k over all stages. Note that martingale tail inequalities are not useful in this setting because they are not sensitive to this global constraint.

As a specific illustration we refine a bicriterion optimization result due to Ravi [6]. Motivated by the Telephone broadcast problem, Ravi gave a polynomial-time algorithm for constructing a spanning tree of small diameter and small maximum degree in a graph G. He showed that, if G has a spanning tree of diameter at most Δ and maximum degree at most D^*, then his algorithm produces a spanning tree of diameter $O(\Delta \log n)$ and maximum degree $O(D^* \log n + log^2 n)$ with high probability. Our analysis of the same algorithm using the gambling game shows that the algorithm produces a spanning tree of diameter $O(\Delta \log n)$ and maximum degree $O(D^* \log n)$ with high probability.

Subsequent to Ravi's paper, Bar-Noy, Guha, Naor and Schieber had also addressed the problem of constructing a short tree of small degree. In [1], they presented an algorithm which constructs a tree of diameter $O(\Delta \log n)$ and maximum degree $O((D^* + \Delta) \log n)$. The algorithm relies on a version of the randomized rounding theorem from [3], which exploits the fact the the sum of the absolute values of the entries of any column of the constraint matrix are small; we note that using the version from [7, 4], which exploits matrices such that columns have few non-zero entries, would still give a bound on the degree that would depend on Δ. (The Telephone broadcast problem now has a much better approximation algorithm [2], but that algorithm is purely combinatorial and no longer relies either on linear programming or on trees of small height and degree; however we consider the bicriteria problem of constructing trees of small height and degree as interesting in its own right).

2 Definitions

Definition 1. *Given a non-negative integer x called the goal and a non-negative real number y called the fortune, a game is defined recursively as follows: if $x = 0$, the game is a win, if $x > 0$ and $y = 0$, the game is lost. If $x, y > 0$, then the game consists of a (finite or infinite) sequence of bets (p_i) such that $\sum_i p_i \leq y$, along with, for each i, a game for goal $x - 1$ and fortune $y - \sum_{j \leq i} p_i$. The success probability of a game is the probability that the game eventually ends up in a winning state.*

A game can be represented by a (possibly infinite) complete binary tree with labelled edges, where the two edges from the root are labelled $1 - p_1$ and p_1, the left child of the root is a game for $(x, y - p_1)$, and the right child of the root is a game for $(x - 1, y - p_1)$.

The success probability can be computed as follows.

Fact 1 *The success probability of a game has the following properties.*
If $x = 0$ then the probability equals 1.
If $x \geq 1$ and $y = 0$ then the probability equals 0.
Otherwise, the success probability of a game T is given by:

$$\Pr(T \text{ succeeds}) = \sum_i \prod_{j<i} (1 - p_j) p_i \Pr(T_i \text{ succeeds}), \tag{1}$$

where p_1 is the first bet of the game, p_i is the i^{th} bet of the game if all previous bets were unsuccessful, and T_i is the remaining game played when the i^{th} bet is the first successful bet.

Note that T_i is a game with goal $x - 1$ and fortune $y - \sum_{j \leq i} p_i$). The (possibly infinite) number of terms in the sum in Equation 1 is the maximum number of bets performed by the game while the goal is x.

Note that as defined, for a given intermediate state (x', y'), the game may decide to bet different amounts, depending on the past history of the game from its starting point. Indeed, if we label vertices of the game tree by the current goal and fortune, there may be several vertices with the same label (x', y'), and each of them is root of a game for (x', y'); these games may all be different from one another.

Definition 2. *A memoryless game is a game such that at every step, the bet placed depends only on the current goal x' and on the current fortune y'. A strategy $\mathcal{H}$ is a function $(x, y) \mapsto p$, where x is a positive integer, y is a positive real number, and $p \in (0, 1]$ is such that $p \leq y$.*

A memoryless game can naturally be extended into a strategy $\mathcal{H}$ by defining $\mathcal{H}(x, y) = \min(1, y)$ for every (x, y) which does not appear as a label of a tree node. Conversely, to any strategy naturally corresponds a game for each (x, y), which proceeds as follows: Consider the current state (x, y). If $x, y > 0$, we place the bet $p = \mathcal{H}(x, y)$. With probability p, the bet is successful and the new goal is $x' = x - 1$; with probability $1 - p$, the bet is unsuccessful and the goal is still $x' = x$. We then continue playing the game associated to $\mathcal{H}$ on the new state $(x', y - p)$.

If $H(x, y)$ denotes the success probability of the game associated to strategy $\mathcal{H}$, Equation 1 then becomes:

$$H(x,y) = \sum_i \prod_{j<i} (1 - p_j) p_i H(x - 1, y - \sum_{j \leq i} p_i), \tag{2}$$

where $p_1 = \mathcal{H}(x, y)$, and in general $p_i = \mathcal{H}(x, y - \sum_{j<i} p_j)$ if $y - \sum_{j<i} p_j$ is positive. When we talk about the success probability of a strategy, we refer to the success probability of the associated game.

Given x and y, we are interested in computing the supremum, over all games for (x, y), of the success probability of the game. Some easy cases may serve as a warmup: evidently this success probability equals 1 whenever $y \geq x$ since it is then sufficient to place x bets each equal to 1. Another easy case is when $x = 1$ and $y < 1$: the supremum is then reached by the strategy which bets y as we now explain.

Lemma 2. $\sup_{T \textit{ game for } (1,y)} \Pr(T \textit{ succeeds}) = \min(1, y)$.

Proof. Given that the fortune is y, the expected number of successes is always at most y, regardless of the game. Thus y is an upper bound to the probability that the number of successes is at least 1. This is reached by the strategy which makes a single bet equal to y. □

Definition 3. *A* continuous strategy *is an extension of strategies which in addition is allowed to place infinitesimal bets, of the form: "repeat betting infinitesimal bets until there is a success or until the fortune spent equals p", for some $p \in (0, y]$.*

We use the notation exp_z to mean the step: "repeat betting infinitesimal bets until there is a success or until the fortune spent equals z". For consistency, a continuous strategy obviously has: If $\mathcal{H}(x, y) = exp_z$, then $\mathcal{H}(x, y - t) = exp_{z-t}$ for every $t \in (0, z]$. This can be seen as the limit, as N tends to infinity, of the process which bets $(1/N, \ldots, 1/N)$ up to zN times or until first success. Since the binomial distribution converges to a Poisson process in the limit, the time to first success is distributed exponentially: for any real number $t \leq z$, we have:

$$\Pr(\text{fortune spent at the end of this step is } \geq t) = e^{-t},$$

and the probability that a success occurs during this step is $1 - e^{-z}$. We will often use the term "discrete strategies" as a synonym for strategies, to contrast them from continuous strategies.

These definitions formalize the Best Strategy Theorem stated in the introduction. Theorem 1, which is our main result, follows from the Best Strategy Theorem as a simple corollary.

3 Proof of the Best Strategy Theorem

This section is devoted to the proof of Theorem 2 for $y < x$ (the Theorem is obvious for $y \geq x$). We will prove that $F(x,y) = G(x,y)$. In subsection 3.1 we will prove that $F(x,y) \leq G(x,y)$. In subsection 3.2, we will prove that $F(x,y) \geq G(x,y)$.

3.1 The upper bound

Definition 4. *A game for (x,y) is finite if its game tree is finite. A strategy is finite if for every (x,y), the associated game is finite.*

Lemma 3. *A game if finite if and only if each tree T_i in Fact 1 is finite, and the number of such trees is finite. A discrete strategy is finite if and only if, for every (x,y), the number of terms in the sum in Equation 2 is finite.*

Proof. The statement of the lemma is obvious for games. As for the statement for strategies, one direction is obvious and the other one can be proved by infuction on x. □

The following lemma shows a reduction from games to finite games.

Lemma 4. *Given x, y, ϵ and a discrete game T for (x,y), there exists a game U for (x,y), which is finite, and such that* $\Pr(T \text{ succeeds}) \leq \Pr(U \text{ succeeds}) + x\epsilon$.

Proof.
Given T, consider the following game U which simulates T.

Game U to simulate the game tree T:
Let (p_j) be the sequence of bets which would be placed by T on (x,y), if every bet was unsuccessful.
If $\sum_j p_j \geq \epsilon$ then place a bet $q = p_1$, and
  if the bet is successful,recursively simulate the game represented by the left subtree;
  if not,recursively simulate the game represented by the right subtree.
Otherwise, play the game associated to the strategy which bets $q = \min(1, y)$.

The proof is by induction on x. Note that $\sum_j p_j$ is at most y, hence the series converges. Let i_0 be the number of terms of the sum if that is finite, or else the smallest index such that $\sum_{j>i_0} p_j < \epsilon$. Game U coincides with T for the first i_0 bets, and makes at most $i_0 + 1$ bets while the goal is x, hence U is a finite game by induction on x and by Lemma 3. A short calculation concludes the proof. □

Fix $\epsilon > 0$. Consider a game T for (x, y). By Lemma 4, there exists a finite game U such that $\Pr(T \text{ succeeds}) \leq \Pr(U \text{ succeeds}) + x\epsilon$. Since this holds for every ϵ, we deduce that $\Pr(T \text{ succeeds}) \leq \sup_{U \text{ finite game}} \Pr(U \text{ succeeds})$. Since this holds for every T, we deduce that $F(x, y) = \sup_{U \text{ finite game}} \Pr(U \text{ succeeds})$.

The following lemma shows a reduction from finite games to finite memoryless games.

Lemma 5. *Given x, y and a finite game T, there exists a finite game U which is memoryless, and such that $\Pr(T \text{ succeeds}) \leq \Pr(U \text{ succeeds})$.*

¿From Lemma 5, we get that $F(x, y) = \sup_{\mathcal{H} \text{ finite strategy}} H(x, y)$. To finish the proof of the upper bound, all we need is to prove the following Proposition, to which we will devote the rest of this section.

Proposition 1. *$\mathcal{G}$ is better than any finite strategy.*

We start by observing that G is convex.

Lemma 6 (Convexity). *If $x \geq y + 1$ then $G(x+1, y) + G(x-1, y) \geq 2G(x, y)$.*

Proof. The proof uses induction on $x + \lceil y \rceil$. The base case $x = 1, y = 0$ is easy. Consider the general case. Run the three processes $G(x+1, y)$, $G(x, y)$ and $G(x-1, y)$ so as to couple the Poisson processes.

Case 1: If $y \leq x - 2$ then all three processes start with an exponential waiting time to first success. We use straightforward induction on G for $x - 1$ and the remaining fortune at the time of first success.

Case 2: If $y \in (x-2, x-1]$ then we let $y' = x - 2$ and observe the three processes until the remaining fortune is y'. For a shorthand, let $G_z = G(z, y')$ for any z. Let $a = y - y'$. After some calculations, we get:

$$G(x-1, y) + G(x+1, y) - 2G(x, y) = e^{-a}(G_{x-1} + G_{x+1} - 2G_x) + ae^{-a}(1 + G_x - 2G_{x-1}).$$

By induction hypothesis for $y' = x - 2$ (noting that $\lceil y' \rceil < \lceil y \rceil$), both quantities within brackets are non negative, hence the lemma.

□

The following is a technical Lemma which will be used in the sequel. It uses the notion of continuous games, similar to the notion of continuous strategies.

Definition 5. *Given (x, y), a continuous game is defined recursively as follows: if $x = 0$, the game is a win, if $x > 0$ and $y = 0$, the game is lost. If $x, y > 0$, then the game consists of a (finite or infinite) sequence of steps, where step i consists either of bet $p_i > 0$ or of the repetition of infinitesimal bets until there is a success or until the fortune spent equals p_i; we must have $\sum_i p_i \leq y$. For each i, if step i was a positive bet p_i, then we also have a game for goal $x - 1$ and fortune $y - \sum_{j \leq i} p_i$; if step i was a sequence of infinitesimal bets up to p_i, then we also have, for each t such that $p_1 + \cdots + p_{i-1} < t \leq p_1 + \cdots + p_i$, a game for $(x - 1, y - t)$.*

Lemma 7. *Let H and K be two continuous games for (x, y) which both go through a state where the remaining fortune is $y' < x - 2$, after having had 0, 1 or 2 successes, and then continue with strategy $\mathcal{G}$ after that point. Then $K(x, y) \geq H(x, y)$ if and only if the probability of having had 0 successes before y' is greater for K than for H.*

Proof. Uses the convexity lemma. □

The following lemma is the core of the proof of Theorem 2.

Lemma 8. *Consider a continuous game whose first bet is arbitrary positive and which then continues by using strategy $\mathcal{G}$. Then its success probability is less than or equal to $G(x, y)$.*

Proof. The proof is by induction on x. If $x = 0$, then there is nothing to prove. Consider $x \geq 1$. Let p be the first bet placed by the game on (x, y). Let $K(x, y)$ denote the success probability of the game. There are several cases.

Case 1:$y \leq x - 1$. Then $\mathcal{G}$ starts by making infinitesimal bets.

Subcase 1.1: Assume $y - p < x - 2$. Let $y' = y - p$. We compare K to the following game L. L makes infinitesimal bets until first success or y'; in the former case, let t be the remaining fortune at that time of first success: L then places a bet of $t - y'$ to get to fortune y'. Once the fortune is y', L continues by following strategy $\mathcal{G}$.

We appeal to Lemma 7 to compare K and L. The probability that K has had 0 successes by the time the fortune is y' is $1 - p$. The probability that L has had 0 successes is $e^{-p} > 1 - p$, hence $L(x, y) > K(x, y)$.

It is now easy to compare L to $\mathcal{G}$: $L(x, y) = \int_0^p e^{-z} L'(x - 1, y - z)dz + e^{-p} G(x, y - p)$. Game L' places a first bet of $(y - z) - y'$ and then continues using strategy $\mathcal{G}$. By induction applied to x and L', we have $L'(x - 1, y - z) \leq G(x - 1, y - z$. Thus $L(x, y) \leq \int_0^p e^{-z} G(x - 1, y - z)dz + e^{-p} G(x, y - p) = G(x, y)$.

Together, these inequalities imply $K(x, y) \leq L(x, y) \leq G(x, y)$.

Subcase 1.2: Assume $y - p \geq x - 2$. Let $y' = x - 2$. Our game K first bets p, bringing its fortune down to $y - p$, then applies $\mathcal{G}$: if the first bet was successful, it bets $r = (y - p) - y'$, bringing its fortune down to y'. If it was unsuccessful, it makes infinitesimal bets until a first success (when the remaining fortune is t) or y', and in the first case, bets $t - y'$, bringing the fortune down to y'.

We compare K to the following game L: makes infinitesimal bets until first success (when the remaining fortune is t') or y', and in the former case, bets $t' - y'$, bringing the fortune down to y'. Once the fortune is y', L continues by following strategy $\mathcal{G}$.

We appeal to Lemma 7 to compare K and L. The probability that K has had 0 successes by the time the fortune is y' is $(1-p)e^{-r}$. The probability that L has had 0 successes is $e^{-(p+r)} = e^{-p}e^{-r} > (1 - p)e^{-r}$, hence $L(x, y) > K(x, y)$.

The comparison of L to G is similar to Subcase 1.1. **Case 2:** $y \geq x - 1$. (Of course, we still have $y < x$). Then $\mathcal{G}$ starts by betting $y - (x - 1)$.

Subcase 2.1: $p < y - (x - 1)$. Let $z = x - 1$. Game K first bets p, bringing its fortune down to $y - p$, then applies $\mathcal{G}$ by betting $r = (y - p) - z$ to bring the fortune down to z, then continues applying $\mathcal{G}$. We compare K to the game associated to strategy $\mathcal{G}$: make a single bet of $y-z$ to bring the fortune down to z, then continue applying $\mathcal{G}$.

The winning probability of K is $K(x,y) = p + (1-p)r + (1-p)(1-r)G(x,z)$. The winning probability of $\mathcal{G}$ is $(p+r)+(1-p-r)G(x,z)$, and one easily checks that $\mathcal{G}$ is better than K.

Subcase 2.2: $p > y-(x-1)$. Let $y' = x-2$. Game K bets p, bringing the fortune down to $y-p$, then, in case of success, bets $(y-p)-y'$; in case of failure, it makes infinitesimal bets until first success (when the remaining fortune is t) or y', and in the former case, bets $t-y'$. Strategy $\mathcal{G}$ first bets $u = y-(x-1)$, then, in case of success, bets 1 to bring the fortune down to y'; in case of failure, it makes infinitesimal bets until first success (when the remaining fortune is t) or y', and in the former case, it bets $t-y'$ ro bring the fortune down to y'.

We appeal to Lemma 7 to compare K and $\mathcal{G}$. The probability that K has had 0 successes is $p_K = (1-u-v)e^{-(1-v)}$, where $u = y-(x-1)$ and $u+v = p$. The probability that $\mathcal{G}$ has had 0 successes is $p_{\mathcal{G}} = (1-u)e^{-1}$. The ratio is

$$\frac{p_K}{p_{\mathcal{G}}} = (1-\frac{v}{1-u})e^v < (1-v)e^v < e^{-v}e^v = 1,$$

hence $\mathcal{G}$ is better than K. $\square$

Proposition 1 then follows by induction on N, the maximum number of steps of the finite strategy applied to (x,y), and by appealing to Lemma 8.

3.2 The lower bound

Lemma 9. *Let T be a continuous game for (x,y). For each positive ϵ, there exists a discrete game U for (x,y) such that* $\Pr(U$ *succeeds*$) \geq \Pr(T$ *succeeds*$) - x\epsilon$.

Proof. We will compare T to the following randomized game U.

Game U to simulate T on (x,y):
If T places a positive bet $p>0$, then bet p;
  if successful, recursively simulate the right subtree of T;
  if not, recursively simulate the left subtree of T.
Otherwise (T places an infinitesimal bet spending up to p), bet $\alpha = \min(p,\epsilon)$;
  if unsuccessful, recursively simulate the game for $(x, y-\alpha)$;
  otherwise, with probability $[e^{-\alpha}-(1-\alpha)]/(1-\alpha)$, recursively simulate the game for $(x,y-\alpha)$; and with the remaining probability, recursively simulate the game for $(x-1, y-T)$, where the random variable $T \in [0,\alpha]$ has density function $e^{-t}/(1-e^{-\alpha})$.

It is easy to see that U simulates T exactly (except for "giving away" a success with probability $(1-\alpha)\times[e^{-\alpha}-(1-\alpha)]/(1-\alpha)$) while spending only a little bit more fortune: every time T has a success, game U spends up to ϵ more fortune than T. But T needs only x successes to reach the goal: so, if T on (x,y) still has a remaining fortune of at least $x\epsilon$ when it reaches its goal, then an initial fortune of y will be sufficient for game U. The probability that T reaches its goal while spending the last $x\epsilon$ part of its fortune, is bounded by the probability that it has one or more success while spending that last $x\epsilon$ fortune; that is less than or equal to the expected number of successes during that time, i.e. less than or equal to $x\epsilon$. Hence $\Pr(U \text{ succeeds}) \geq \Pr(T \text{ succeeds}) - x\epsilon$.

Finally, it is easy to de-randomize U: just pick some T' such that the success probability for the game for $(x-1, y-T')$ is greater than or equal to the expected value, over T, of the game for $(x-1, y-T)$. We thus obtain a strategy satisfying the Lemma. □

¿From Lemma 9, we get that $F(x,y) = \sup_{T \text{ continuous game}} \Pr(T \text{ succeeds})$, which is obviously greater than or equal to $G(x,y)$, and the proof of Theorem 2 is complete.

4 A randomized rounding application

In [6], Ravi presented an algorithm to build a spanning tree of small diameter and small maximum degree in a given graph. Here, using the framework of gambling games, we present a finer analysis of Ravi's algorithm, thus improving on his approximation bounds. Here is the algorithm.
Input: a graph G with vertex set $V(G)$ and a bound Δ on the desired diameter.
Output: a spanning tree G

Dynamic variables in the algorithm are a subgraph K of G and a set $C \subseteq V$ of cluster centers.

1. Initialize K to a graph with vertex set $V(G)$ and no edges; Initialize C to $V(G)$.
2. While there is more than one cluster center do:
   (a) Set up an integer program of the type described in the Introduction, where:
      i. For every path P of length at most Δ directed from one cluster center to another there is a 0-1 variable $x(P)$;
      ii. For every cluster center $c \in C$ there is a covering constraint of the form $\sum x(P) = 1$, where the sum is over all paths P directed out of c;
      iii. For every vertex v there is a resource constraint of the form $\sum x(P) \leq z$, where the sum is over paths P incident with vertex v.
   (b) Solve the linear programming relaxation of this integer program (Ravi shows that this can be done in polynomial time);
   (c) Use randomized rounding to obtain a feasible solution to the integer program, giving, for each cluster center, a path of length at most Δ directed to some other cluster center;

(d) Consider the graph H with one vertex for each cluster center in C and one directed edge (c, c') for each path in the solution to the integer program. Each vertex in H has out-degree exactly 1. By elementary graph theory, find a subgraph H' of H containing at least $\frac{|V(H)|}{3}$ edges and consisting of a disjoint union of "stars," where each star consists of a root vertex c' and one or more vertices c such that (c, c') is an edge of H;

(e) For each edge (c, c') in H', add the corresponding path to K, and delete c from C, the set of cluster centers.

3. Let c be the unique remaining cluster center. By breadth-first search from c in K, construct a tree T spanning $V(G)$.

Theorem 3 (Ravi). *Assume that G has a spanning tree T^* of diameter at most Δ and maximum degree at most D^*. Then with high probability the above algorithm will produce a spanning tree of height $O(\Delta \log n)$ and maximum degree $O(D^* \log n + \log^2 n)$.*

Proof. The height of T equals the height of K. Since the "while" loop is executed $O(\log n)$ times, each vertex $v \in G$ is linked to c in K by a sequence of at most $O(\log n)$ flow paths, each of length at most Δ. Hence T has height $O(\Delta \log n)$. Using T^*, it is easy to construct a multicommodity flow of length at most Δ and value at most $2D^*$. Hence the solution of the LP in step 2a has value at most $2D^*$. By the randomized rounding Theorem (which is based on a Hoeffding bound), the integral multicommodity flow in step 2b has value at most $2D^* + O(\log n)$ with high probability, and so the union of the flow paths taken in step 2e also has maximum degree at most $2D^* + O(\log n)$. Since the "while" loop is executed $O(\log n)$ times, the resulting graph K has maximum degree $O(D^* \log n + \log^2 n)$ (with high probability), and hence the output T also has maximum degree $O(D^* \log n + \log^2 n)$ (with high probability). □

We will use our gambling game to provide a more refined analysis of Ravi's algorithm.

Theorem 4. *Assume that G has a spanning tree T^* of diameter at most Δ and maximum degree at most D^*. Then the above algorithm will produce a spanning tree of height $O(\Delta \log n)$ and maximum degree $O(D^* \log n)$ (with high probability).*

Proof. Fix a vertex ℓ of G. We play the gambling game: the initial fortune is $2D^*t$ where t is the number of integer programs solved in the algorithm, and the goal is $6\alpha D^* \log n$, where α is chosen to guaranteed that $t \leq \alpha \log n$. The bets are done in phases corresponding to the successive integer programs in the algorithm. In each phase there is a bet for each cluster center, equal to the probability that vertex ℓ will lie in the path from that cluster center selected by the integer program. The sum of these bets is just the sum of the fractional variables in the linear program corresponding to directed paths passing through ℓ. Since the value of the linear program is at most $2D^*$ the sum of the bets in each phase is at most $2D^*$ and the sum of all bets does not exceed the fortune $2D^*t$. The degree of ℓ in the tree T is at most twice the number of selected paths through ℓ in the course of the algorithm, and the number of selected paths is equal to the number of successes in the gambling game. Hence the probability that the degree of ℓ is T is greater than or equal to $6\alpha D^* \log n$ is at most $F(3\alpha D^* \log n, 2\alpha D^* \log n)$, which can be shown to be exponentially small in n. $\square$

References

1. Amotz Bar-Noy, Sudipto Guha, Joseph (Seffi) Naor, and Baruch Schieber, *Multicasting in Heterogeneous Networks*, SIAM J. Comput. 30(2): 347-358 (2000).
2. Michael Elkin and Guy Kortsarz, *Sublogarithmic Approximation for Telephone Multicast: Path out of Jungle*, SODA 2003, to appear.
3. R. M. Karp, F. T. Leighton, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani, *Global wire routing in two-dimensional arrays*, Algorithmica, 2, 113-129, 1987.
4. New Algorithmic Aspects of the Local Lemma with Applications to Routing and Partitioning, F. T. Leighton, C.-J. Lu S. B. Rao and Aravind Srinivasan. SIAM Journal on Computing, Vol. 31, 626-641, 2001.
5. P. Raghavan and C.D. Thompson, *Randomized rounding: a technique for provably good algorithms and algorithmic proofs*, Combinatorica 7 (1987), 365-374.
6. R. Ravi, *Rapid Rumor Ramification: Approximation the minimum broadcast time*, 35^{th} IEEE Symposium on Foundations of Computer Science (FOCS), 1994.
7. Aravind Srinivasan, *An Extension of the Lovasz Local Lemma and its Applications to Integer Programming*, Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 6-15, 1996.