Look Up and Scream: Analytical Difficulties Resolved!

Swapnil P. Bhatia
Dept. of Computer Science
Univ. of New Hampshire
N229, Kingsbury Hall, Durham NH, 03824
Email: sbhatia@cs.unh.edu

Abstract

We solve a counting problem proposed by Balas and Connor [1] derived from a game called *Look Up and Scream*. Our solution applies to both variants of the game and is based on counting matchings of every size in the constraints graph of the game. We count such matchings directly in the unconstrained variant of the game and use a counting technique discovered by J. P. McSorley [2] for the constrained variant.

1 Look Up and Scream

Balas and Connor introduce a new game called *Look Up and Scream (LUaS)* [1]. The game involves $2N$ participants standing in a circle. The game is played in rounds where each round proceeds as follows: At the beginning of the round, every participant looks down at the floor. Then, on cue, all the participants look up simultaneously, and instantaneously choose to fix their gaze on another player. If any two players realize that they have chosen to gaze at each other, they scream and the round ends with a *loss* or *failure*. A new round is then played. The game ends if a round is silent, i.e., no players choose to gaze at players who are gazing back at them. The rules as outlined above describe the unconstrained variant of LUaS. In the constrained variant, a player can only choose to gaze at either her neighboring players or the player situated at the antipode of her location. Obviously, many other variants can be concieved, one for each subset of players that any player is allowed to target. Balas and Connor wish to determine the probability of winning in any round in a game of LUaS. They present numerical solutions for the problem obtained by exhaustive search but do not provide any formula for determining the probability analytically. Further, the authors make the following comment about the problem [1]:

> "The approach of attempting to enumerate the number of gazes available to each player is complicated by the non-local properties [. . .] At present, we have discussed this problem with many mathematicians (both recreational

Figure 1: An example of the constraints graph of the LUaS game with 6 players. The labeled vertices indicate players and the edges indicate possible (bidirectional) gaze choices for each player.

> and otherwise) and have been unable to determine any general method for counting the number of winning gaze arrangements in a $2N$-player game of LUaS. We invite the readers of this journal to contribute their talents and their time to help us determine if such a method exists."

In this note, we demonstrate a method for counting the number of losing configurations for both the constrained and the unconstrained variant of LUaS, from which, the number of winning configurations, and hence the winning probability, can be easily obtained. Counting in the unconstrained variant is direct and straightforward. For the constrained variant, we use a counting technique discovered by J. P. McSorley [2].

2 Approach

We denote the set of integers $\{1, \dots, k\}$ by $[k]$. We define $LUaS(2N; \{s_1, \dots, s_m\})$ as a game of LUaS involving $2N$ players labeled $\{0, \dots, 2N-1\}$ and $s_j \in [N]$ with each player i having the choice to gaze at any other player $p \in \{i \pm s_j \mod 2N\}$. We restrict our discussion to the following two variants proposed in [1]: the constrained variant $LUaS(2N; \{1, N\})$ and the unconstrained variant $LUaS(2N; [N])$.

We observe that the rules of a particular LUaS game, i.e., constraints on the possible gaze choices for any player, can be represented in the form of a graph on $2N$ vertices, which we call the *constraints graph* of that particular LUaS variant. The constraints graph of a LUaS variant $LUaS\{2N; \{s_1, \dots, s_m\}\}$ is an undirected graph $G(V, E)$ on the set of vertices $V = \{0, \dots, 2N-1\}$ with the set of edges

$$E = \{\{i, j\} : i, j \in V, i \neq j, \exists k \in [m]\ (j = i \pm s_k \mod 2N)\}. \qquad (1)$$

We denote by E_d the directed version of E, i.e., $E_d = \{(i, j) : \{i, j\} \in E\}$. Fig. 1 shows an example ($N = 3$) of the constraints graph of both LUaS variants. We refer to the collection of $2N$ gazes in a realization of a round of the LUaS game as a *configuration*. That is, a configuration is a set of pairs $Z \subseteq E_d$ such that $\forall u \in V$ there exists a unique $v \in V$ such that $(u, v) \in Z$. A *sub-configuration* is any subset of a configuration. By definition of the LUaS game, a *losing configuration* Z is a configuration in

Figure 2: Examples of winning and losing configurations.

which $(u, v) \in Z$ and $(v, u) \in Z$. The pair $\{u, v\}$ is called a *losing pair*. A *winning configuration* is one with no losing pair. Fig. 2 shows an example of a winning and losing configuration for each variant of the game. We consider synonymous any losing pair $\{u, v\}$ in a losing configuration Z and edge $\{u, v\} \in E$, which we call a *losing edge*. A *losing set* is a set of losing edges $L \subset E$ such that $\{(u, v) : \{u, v\} \in L\}$ is a sub-configuration. $\mathcal{L}(Z)$ is the set of all losing edges of configuration Z. Given a losing set L of size s, suppose we wish to construct a configuration Z such that $L \subseteq \mathcal{L}(Z)$. We could achieve this using algorithm $LosingPairs(L, G)$ (LP) outlined below:

```
ALGORITHM $LosingPairs(L, G(V, E))$:

1. Initialize configuration $Z \leftarrow \emptyset$ and set $V' \leftarrow \emptyset$.
2. For each given losing edge $e = \{u, v\} \in L$:
    (a) Add $(u, v)$ and $(v, u)$ to $Z$.
    (b) Add vertices $u$ and $v$ to $V'$.
3. For each vertex $x \in V \setminus V'$:
    (a) Select an edge $\{x, y\} \in E$.
    (b) Add $(x, y)$ to $Z$.
```

We shall ignore the order of items in L and identify a *distinct run of LP* by its choice of an edge in step (3a). (Again, order of edge selection will not matter.) It is easy to verify that each distinct run of algorithm LP will produce a configuration Z such that input $L \subseteq \mathcal{L}(Z)$. All distinct runs (as defined above) of LP taken together will generate all the different configurations Z such that $L \subseteq \mathcal{L}(Z)$. Now let C_i denote the set of losing configurations in which the i-th edge of G, denoted by e_i, is a losing pair. We can generate all the configurations in C_i by collecting the configuration Z from all the distinct runs of $LosingPairs(\{e_i\}, G)$.

Let $M(N, s)$ denote the number of ways to choose a set L of s losing edges and let $F(N, s)$ denote the number of ways in which step (3a) of the algorithm LP can be performed (which is also the number of distinct executions of LP for a given input). From the above discussion it is clear that:

$$\sum_i |C_i| = M(N, 1) \cdot F(N, 1), \tag{2}$$

since $i \in [M(N, 1)]$ and for each value of i, LP can be executed in $F(N, 1) = |C_i|$ distinct ways. Similarly, executing $LosingPairs(\{e_i, e_j\}, G)$ shows:

$$\sum_{i,j} |C_i \cap C_j| = M(N, 2) \cdot F(N, 2), \tag{3}$$

and in general, executing $LosingPairs(\cup_{j\in[s]}\{e_{i_j}\}, G)$ shows:

$$\sum_{i_1, i_2, \cdots, i_s} \left| \bigcap_{p\in[s]} C_{i_p} \right| = M(N, s) \cdot F(N, s). \tag{4}$$

Then, by the principle of Inclusion-Exclusion:

$$\begin{aligned} \left| \bigcap_i \bar{C}_i \right| &= |U(N)| + \sum_{k\in[N]} (-1)^k \left(\sum_{i_1, i_2, \cdots, i_k} \left| \bigcap_{p\in[k]} C_{i_p} \right| \right) &(5)\\ &= |U(N)| + \sum_{s\in[N]} (-1)^s M(N, s) \cdot F(N, s). &(6) \end{aligned}$$

The set $\bigcap_i \bar{C}_i$ denotes the set of configurations in which no edge is a losing edge, i.e., the set of winning configurations, and the set $|U(N)|$ is the set of all distinct configurations compatible with the constraints graph of a LUaS game with $2N$ players. The ratio of the cardinality of this set to the total number of configurations is the required probability of winning in any round of LUaS. What remains is to measure the cardinality of each of the sets $\sum_i |C_i| = M(N, 1) \cdot F(N, 1)$, $\sum_{i,j} |C_i \cap C_j| = M(N, 2) \cdot F(N, 2)$, and so on for the constrained and unconstrained variants.

2.1 Gaze Choices and $F(N, s)$

Let $C_u(2N)$ and $C_c(2N)$ denote the number of gaze choices available to a player in the unconstrained and constrained variants of LUaS respectively. Then:

$$\begin{aligned} C_u(2N) &= 2N - 1, \text{ and} &(7)\\ C_c(2N) &= 3. &(8) \end{aligned}$$

Then, since the selection of an edge for each vertex in step (3) of LP is independent, we can write:

$$\begin{aligned} F_u(N,s) &= C_u(2N)^{(2N-2s)}, \text{ and} && (9)\\ F_c(N,s) &= C_c(2N)^{(2N-2s)}. && (10) \end{aligned}$$

2.2 Matchings and $M(N,s)$

A *matching* of an undirected graph G is a subset of its edges such that all edges are pairwise disjoint. We define an *s-matching* to be a matching with exactly s edges. Observe that choosing a valid losing set L of size s requires us to select an s-matching of the constraints graph G. Thus, to calculate $M(N,s)$, we must calculate the number of s-matchings in the constraints graph of the LUaS variant under consideration.

2.3 Total Configurations U(N)

Let $U_u(N)$ and $U_c(N)$ denote the set of all distinct configurations compatible with the constraints graphs of $LUaS(2N;[N])$ and $LUaS(2N;\{1,N\})$ respectively. Clearly,

$$\begin{aligned} |U_u(N)| &= C_u(2N)^{2N}, \text{ and} && (11)\\ |U_c(N)| &= C_c(2N)^{2N}. && (12) \end{aligned}$$

3 Unconstrained LUaS: $M_u(N,s)$

We first consider the unconstrained variant, $LUaS(2N;[N])$, of the game. To construct an s-matching, we use algorithm $UnconstrainedMatching(N,s)$ (UM) below:

```
ALGORITHM $UnconstrainedMatching(N,s)$:

1. Initialize $V \leftarrow \{0,\dots,2N-1\}$ and matching $M \leftarrow \emptyset$.
2. Repeat the following steps $s$ times:
    (a) Choose a pair of vertices $\{u,v\} \in V$ and set $M \leftarrow M \cup \{u,v\}$.
    (b) Update $V \leftarrow V \setminus \{u,v\}$.
```

Each iteration of the loop in algorithm UM, adds one pair to the matching. The addition step (2a) can be performed in $\binom{|V|}{2}$ ways. Initially $|V| = 2N$ and $|V|$ decreases by 2 in each iteration. However, since the order in which the s-matching is generated does not matter, we must divide by $s!$. Hence, the total number of distinct s-matchings generated will be:

$$M_u(N,s) = \prod_{k=1}^{s} \frac{1}{k}\binom{2N-2(k-1)}{2} = \frac{(2N)!}{(2N-2s)!\; s!\; 2^s}. \qquad (13)$$

Figure 3: The constraints graph of $LUaS(2N; \{1, N\})$ is isomorphic to the Möbius ladder of order N. The constraints graph on the right illustrates the terminology used to count matchings using McSorley's technique [2].

Substituting (9), (11), and (13) in (6) and simplifying gives,

$$\left| \bigcap_{i \in [2N]} \bar{C}_i \right| = (2N-1)^{2N} \left(1 + (2N)! \sum_{s=1}^{N} \frac{(-1)^s}{2^s (2N-1)^{2s}} \cdot \frac{1}{s!(2(N-s))!} \right). \quad (14)$$

Therefore, the probability of winning in any round of $LUaS(2N; [N])$ is given by:

$$P^u_{win} = \sum_{s=0}^{N} \frac{(-1)^s}{(2N-1)^{2s}} \cdot \binom{2N}{2s} \cdot (2s-1)!!, \quad (15)$$

where n!! denotes the double-factorial function (with (-1)!! = 1).

4 Constrained LUaS: $M_c(N, s)$

A Möbius ladder is a graph on $2N$ vertices $V = [2N]$ and can be visualized as a pair of chains of N vertices with an additional edge across each corresponding pair of vertices, one from each chain (i.e., the "rungs of the ladder"). The two ends of the ladder are connected like the ends of a Möbius strip. The constraints graph of $LUaS(2N; \{1, N\})$ is isomorphic to the Möbius ladder, as illustrated by the mapping for $N = 4$ in Fig. 3. Since we need to calculate $M_c(N, s)$, we need to count the number of s-matchings in the Möbius ladder. In [2], McSorley provides a technique for counting various structures in the Möbius ladder, including the total number of matchings. However, here we are required to count matchings of a given size s in a Möbius ladder. Nonetheless, we are able to use McSorley's technique provided we can count s-matchings in a chain with r edges, which is straightforward.

We generally follow the notation in [2] and only describe it briefly here. (See Fig. 3 for examples.) We call the $2N$ edges forming the circle of the constraints graph as outside edges and the remaining N edges as diagonals. In any subset of edges of the Möbius ladder, vertices on a diagonal will be referred to as diagonal vertices. The subgraph between two consecutive diagonal vertices (i.e., the chain of outside edges) is known as a *join*. The size of the join is the maximum number r of edges in it. We denote a k-*composition* of an integer N by $x(k) = (x_1, \cdots, x_k)$ where $x_1 + \cdots + x_j + \cdots + x_k = N$ and $\forall j\ x_j \geq 1$. Any subset of k diagonals describes a k-composition of N where the $x_1, \ldots, x_k$ are the sizes of the induced joins. Let $\delta_y(r)$ denote the generating function of the number of i-matchings in a join of size r. Observe that $\delta_y(0) = 0 \cdot y^0$, $\delta_y(1) = 1 \cdot y^0$, and

$$\delta_y(r) = \delta_y(r-1) + y \cdot \delta_y(r-2). \tag{16}$$

Solving this recurrence gives:

$$\delta_y(r) = \frac{1}{\sqrt{1+4y}} \cdot \left[\left(\frac{1+\sqrt{1+4y}}{2} \right)^r - \left(\frac{1-\sqrt{1+4y}}{2} \right)^r \right] \tag{17}$$

Before we generate any s-matching in a Möbius ladder, we first generate one comprising k diagonals and $s-k$ outside edges. If $k=0$, then we are required to generate an s-matching in the cycle of $2N$ vertices, also known as a *nonconsecutive cyclic set*. The number of such matchings is [3]:

$$NCC(2N, s) = \frac{2N}{2N-s} \binom{2N-s}{s}. \tag{18}$$

For $k > 0$, we use the algorithm $ConstrainedMatchingk(N, s, k)$ (CMk) outlined below to generate an s-matching with k diagonals:

```
ALGORITHM $ConstrainedMatchingk(N, s, k)$:

1. Choose vertex 0 to be a diagonal vertex and place a diagonal at vertex 0.
2. Of the remaining $N-1$ vertices, choose $k-1$ vertices to be diagonal vertices
   and place a diagonal at each of them.
3. Distribute the remaining $s-k$ edges as outside edges in the $2k$ joins.
```

If we count the number of distinct executions of CMk, then we can obtain the number of s-matchings with exactly $k > 0$ diagonals, with the first diagonal on vertex 0. We make a few useful observations. The diagonal placed in step (1) along with the $k-1$ diagonals placed in step (2) of CMk will describe a k-composition $x(k) = (x_1, \ldots, x_j, \ldots, x_k)$ of N. This will create joins of sizes x_j over vertices $\{0, \ldots, N-1\}$ as well as a second set of (antipodal) joins of sizes x_j over vertices $\{N, \ldots, 2N-1\}$. The number of ways to distribute i outside edges in a join of size r (to obtain an i-matching) is $[y^i]\delta_y(r)$ where $[y^i]p$ denotes the coefficient of y^i in the polynomial p. Therefore, the number of ways to distribute a total of i outside edges over two antipodal joins, each of size x_j, to obtain an i-matching is $[y^i]\delta_y(x_j)^2$. Hence, the number of ways to distribute $s-k$ outside edges over k pairs of antipodal joins of sizes $x(k) = (x_1, \ldots, x_j, \ldots, x_k)$ to obtain an $s-k$-matching is

$$[y^{s-k}] \prod_{x_j} \delta_y(x_j)^2, \tag{19}$$

Figure 4: Probability of winning in any round in $LUaS(2N;[N])$ and $LUaS(2N;\{1,N\})$.

which is therefore the number of ways to execute step (3) of CMk given a specific choice of diagonals in step (2) of CMk. Summing over different k-compositions $x(k)$ with a diagonal on vertex 1 will give the number of different executions of CMk. The total number of s-matchings can be obtained by multiplying (19) by y^k to account for the k diagonals that also add to the size of the matching. Furthermore, we can choose the first diagonal vertex in N different ways. However, we will have counted each k-composition k times if we were to use the CMk algorithm to generate the s-matching. Thus, the final form of the generating function for the number of matchings of size s in a Möbius ladder will be:

$$m_c(N,s) = NCC(2N,s) + [y^s]\left[\sum_{k=1}^{s} \frac{N}{k} y^k \sum_{x(k)} \prod_{x_j} \delta_y(x_j)^2\right] \tag{20}$$

Using Theorem C (i) in [2], we can write:

$$\sum_{x(k)} \prod_{j=1}^{k} \delta(x_j)^2 = [z^n](\delta(1)^2 z + \delta(2)^2 z^2 + \cdots + \delta(N)^2 z^N)^k. \tag{21}$$

Therefore, we can finally write:

$$M_c(N,s) = NCC(2N,s) + [y^s]\left[\sum_{k=1}^{s} y^k \frac{N}{k} [z^N](\delta(1)^2 z + \cdots + \delta(N)^2 z^N)^k\right]. \tag{22}$$

Substituting (10), (22) and (12) into (6) and dividing by $|U_c|$ will give the required probability. We used the Maple code listed in the Appendix to calculate this probability for specific values of N. Fig. 4 shows the probability calculated for $N \in [15]$ which matches the results in [1].

5 Some Applications of LUaS

5.1 Capacity of directional wireless networks

A realization of the LUaS game occurs in a wireless network with directional antennas. The scenario is as follows. Consider a wireless network with $2N \geq 2$ nodes each equipped with an antenna that can receive from and transmit in exactly one direction at any given time. Thus, communication between two nodes requires that both nodes point their antennas towards each other and only one of them transmit. This situation is clearly analogous to the LUaS game in that a successful communication occurs only in a losing round. The probability of a losing round and the average number of losing pairs of players in any round would provide an estimate of the average "concurrent" capacity of such a network as defined by Balakrishnan et al. [4]. In their paper, a notion of maximum *concurrent* capacity of a wireless network is defined as the maximum number of simultaneous transmissions that can occur in the network which is an estimate of the maximum capacity of the network. Balakrishnan et al. consider a network of omni-directional antennas whereas the LUaS game model in this paper applies to directional networks. $LUaS(2N; [N])$ corresponds to a wireless network where any two nodes are within transmission range of each other. Other variants of the game including $LUaS(2N; \{1, N\})$ correspond to a network scenario in which only certain nodes may be able to transmit to each other. This may be realistic constraint in a wireless network and could be used to model certain routing conditions or channel impairments.

5.2 Peer-to-peer barter networks

The utility of peer-to-peer (P2P) systems such as Napster, KaZaA and Gnutella to its users is diminished due to a large proportion of *free-riders*; these are self-centered non-cooperative users who consume free resources provided by the P2P system but do not contribute any of their own resources to the system. Anagnostakis and Greenwald [5] propose a barter-based design of a P2P system. In their design, a peer p_i shares its resource r_i with another peer p_j if and only if peer p_j has some resource r_j, which it is ready to offer to p_i, and which peer p_i wishes to consume. Clearly, a useful transaction occurs in this P2P system only under such a *double coincidence of wants*. Thus, the probability of such a double coincidence is an estimate of the number of rounds in which successful trades occur and the average number of such double coincidences is the average number of successful transactions. In the context of the LUaS game, a successful P2P transaction in a barter-based P2P system corresponds to a losing game. Anagnostakis and Greenwald also allow for k-way barters which correspond to k-cycles in the LUaS game.

Acknowledgments

The author thanks Dr. Radim Bartoš and Jeff Laird for their careful reading and invaluable comments. The author also thanks Dr. E. Deutsch and Dr. J. P. McSorley for providing the author with a copy of [2]. At the time of this work, the author was supported by the Cisco University Research Program.

References

[1] B. J. Balas and C. W. Connor, "Look Up and Scream: Analytical Difficulties in Improv Comedy," *Journal of Recreational Mathematics*, vol. 33, no. 1, pp. 32–38, 2004-2005.

[2] J. P. McSorley, "Counting structures in the Möbius ladder," *Discrete Mathematics*, vol. 184, pp. 137–164, 1998.

[3] R. P. Stanley, *Enumerative Combinatorics*. Wadsworth & Brooks/Cole, 1986, vol. 1, ch. 2.

[4] H. Balakrishnan, C. L. Barrett, V. S. A. Kumar, M. V. Marathe, and S. Thite, "The Distance-2 Matching Problem and Its Relationship to the MAC-Layer Capacity of Ad Hoc Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1069–1079, Aug. 2004.

[5] K. G. Anagnostakis and M. B. Greenwald, "Exchange-based Incentive Mechanisms for Peer-to-Peer File Sharing," in *24th IEEE International Conference on Distributed Computing Systems (ICDCS 2004)*, Tokyo, Japan, Mar. 2004, pp. 524–533.

Appendix

The following Maple code generates the probability of winning in $LUaS(2N;\{1,N\})$.

$$\begin{aligned}
&joinmatch := r \to (1/2+1/2*sqrt(1+4*y))^r/sqrt(1+4*y)\\
&-(1/2-1/2*sqrt(1+4*y))^r/sqrt(1+4*y);\\
&jm := r \to joinmatch(r);\\
&ThmC := (N,k) \to (sum(jm(r)^2*z^r, r=1..N))^k;\\
&matchkdiag := (N,k) \to (N/k)*\\
&simplify(expand(y^k*coeff(ThmC(N,k),z^N)));\\
&Ncc := (N,s) \to 2*N*binomial(2*N-s,s)/(2*N-s);\\
&matchings := (N,s) \to add(matchkdiag(N,q), q=1..s);\\
&Choices := (N,s) \to 3^{2*(N-s)};\\
&P := N \to evalf((Choices(N,0)+add((-1)^s\\
&*Choices(N,s)*(simplify(Ncc(N,s))+coeff(matchings(N,s),\\
&y^s)), s=1..N))/Choices(N,0));
\end{aligned}$$