NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

NEW FICTITIOUS PLAY PROCEDURE FOR SOLVING BLOTTO GAMES

by

Moon Gul Lee

December 2004

Thesis Advisor:	James N. Eagle
Thesis Co-Advisor:	W. Matthew Carlyle
Second Reader:	Jae-Yeong Lee

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: New Fictitious Play Procedure For Solving Blotto Games			**5. FUNDING NUMBERS**
6. AUTHOR(S) Lee, Moon Gul, CPT ROKAF			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT **Approved for public release, distribution is unlimited**			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

In this thesis, a new fictitious play (FP) procedure is presented to solve two-person zero-sum (TPZS) Blotto games. The FP solution procedure solves TPZS games by assuming that the two players take turns selecting optimal responses to the opponent's strategy observed so far. It is known that FP converges to an optimal solution, and it may be the only realistic approach to solve large games. The algorithm uses dynamic programming (DP) to solve FP subproblems. Efficiency is obtained by limiting the growth of the DP state space.

Blotto games are frequently used to solve simple missile defense problems. While it may be unlikely that the models presented in this paper can be used directly to solve realistic offense and defense problems, it is hoped that they will provide insight into the basic structure of optimal and near-optimal solutions to these important, large games, and provide a foundation for solution of more realistic, and more complex, problems.

14. SUBJECT TERMS Fictitious Play , New FP Procedure, Two Person Zero Sum, Blotto Game, Dynamic Programming			**15. NUMBER OF PAGES** 53
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

NEW FICTITIOUS PLAY PROCEDURE FOR SOLVING BLOTTO GAMES

Moon Gul Lee
Captain, Republic of Korea Air force
B.S., Korea Air Force Academy, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
December 2004

Author:	Moon Gul Lee
Approved by:	Dr. James N. Eagle Thesis Advisor
	Dr. W. Matthew Carlyle Co-Advisor
	Dr. Jae-Yeong Lee Second Reader
	Dr. James N. Eagle Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, a new fictitious play (FP) procedure is presented to solve two-person zero-sum (TPZS) Blotto games. The FP solution procedure solves TPZS games by assuming that the two players take turns selecting optimal responses to the opponent's strategy observed so far. It is known that FP converges to an optimal solution, and it may be the only realistic approach to solve large games. The algorithm uses dynamic programming (DP) to solve FP subproblems. Efficiency is obtained by limiting the growth of the DP state space.

Blotto games are frequently used to solve simple missile defense problems. While it may be unlikely that the models presented in this paper can be used directly to solve realistic offense and defense problems, it is hoped that they will provide insight into the basic structure of optimal and near-optimal solutions to these important, large games, and provide a foundation for solution of more realistic, and more complex, problems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This work would not have been possible without the enormous support of these people. First of all, I would like to extend my deepest appreciation to my thesis advisor, Professor James Eagle, Co advisor, Professor Carlyle Matthew and Second Reader, KNDU Professor Lee Jae-Yeong. Their expert advice, directions and patience have expanded and enhanced my knowledge, and allowed me to complete my thesis.

Additionally, I wish to thank my wife, Cho Jung Hwa and my lovely son Yong Bin for supporting me during the long hours I have spent away from home while finishing this thesis. They have been the greatest supporters and inspirers whenever I was discouraged. In addition, I greatly appreciate my country, the Republic of Korea, its Airforce and KNDU for all the support during my journey to the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Fictitious play (FP), first introduced by Brown and Robinson (1951), is an iterative procedure used to approximate solutions to two-person zero-sum (TPZS) games. At each iteration of FP, each player chooses a pure strategy that is a best reply to the mixed strategy represented by the aggregation of all of other player's pure strategies played so far, assuming they will be chosen based on the empirical probability distribution induced by their historical frequency in all previous iterations. Fictitious play can be thought of as mimicking the behavior of players learning from their opponents.

The purpose of this thesis is to investigate the use of fictitious play in the solution of Blotto games and their generalizations. In Blotto games, opponents each allocate a limited number of forces to a specified number of areas. Payoffs in each area accrue to the players based on the number of forces assigned to each area. The main application of Blotto games has been the analysis of missiles attack and defense problems.

In this thesis a new fictitious play (FP) procedure is presented to solve two-person zero-sum (TPZS) Blotto games. The FP solution procedure solves TPZS games by assuming that the two players take turns selecting optimal responses to the opponent's strategy observed so far. It is known that FP converges to an optimal solution, and it may be the only realistic approach to solve large games. The algorithm we develop uses dynamic programming (DP) to solve the FP subproblems. Efficiency is obtained by limiting the growth of the DP state space. We derive the dynamic programming recurrence relation for solving Blotto games with a general payoff function using fictitious play. The recurrence can be solved without explicitly keeping track of every attack or defense played; rather, the information required is simply the number of times a given force level (number of attackers or defenders) has been used in each area, over all attacks and defenses seen so far. Although our experiments considered one type of attacker and one type of defender, we indicate how to generalize this procedure to cases with more than one type of attacker or defender (or both).

During this study, we identified other topics for further investigations. The first is to investigate generalizations of Blotto games in which defenders can be placed in such a way as to defend multiple areas at once. This is closer to the real situation with missile defense. The second is to explore the issue of playability in the ILP formulations. Our proposed playability constraint is currently too restrictive; we have provided examples in which the optimal solution to the ILP, with the playability constraint, is not equal to the value of the game. Further research should explore less restrictive, alternate formulations of playability constraints. It is possible (although unlikely) that less restrictive playability constraints would also yield more efficiently solvable ILP, making that approach competitive with the DP-based procedure for larger problems.

While it is unlikely that the models presented in this paper can be used directly to solve realistic offense and defense problems, it is hoped that they will provide insight into the basic structure of optimal and near optimal solutions to these important, large games.

I. INTRODUCTION

Fictitious play (FP), first introduced by Brown and Robinson (1951), is an iterative procedure used to approximate solutions to two-person zero-sum (TPZS) games. At each iteration of FP, each player chooses a pure strategy that is a best reply to the mixed strategy represented by the aggregation of all of other player's pure strategies played so far, assuming they will be chosen based on the empirical probability distribution induced by their historical frequency in all previous iterations. Fictitious play can be thought of as mimicking the behavior of players learning from their opponents.

The purpose of this thesis is to investigate the use of fictitious play in the solution of Blotto games and their generalizations. In Blotto games, opponents each allocate a limited number of forces to a specified number of areas. Payoffs in each area accrue to the players based on the number of forces assigned to each area. The main application of Blotto games has been the analysis of missile attack and defense problems.

This thesis is organized as follows: in Chapter II, we introduce TPZS games, Blotto games and the solution procedure. In Chapter III, we solve various versions of the problem using FP. Chapter IV provides conclusions and suggests further work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. TWO PERSON ZERO-SUM GAMES

A. DEFINITION

A two-person zero-sum (TPZS) game (von Neumann and Morgenstern, 1944) is a situation where there are two players having directly opposite interests. In a TPZS game, player 1 (also called *X*, the row player, or the maximizer) has *m* pure strategies and player 2 (*Y*, column player, minimizer*)* has *n* pure strategies. A player can commit to playing a pure strategy, or, by randomizing his choice among several pure strategies, he can employ a *mixed strategy*. A mixed strategy is represented by a vector of probabilities of choosing each pure strategy. For player 1, we write this vector as:

$$x = (x_1, \ldots, x_m)^T .$$

Because *x* is a vector of probabilities, we have the restrictions that

$$\sum_{i=1}^{m} x_i = 1$$

and

$$x_i \geq 0, \qquad i = 1, \ldots, m \ .$$

Similar notation and restrictions are used for player 2, whose mixed strategy is written as:

$$y = (y_1, \ldots, y_n)^T .$$

In a TPZS game, each player chooses a strategy (pure or mixed), unknown to the other, and both strategies are revealed simultaneously. The result of the game depends on the strategy used by each player. If *X* and *Y* choose their i^{th} and j^{th} pure strategies, respectively, then the result of game, denoted a_{ij}, represents the amount that *Y* has to pay *X*. Equivalently, the payoffs to *X* and *Y* are a_{ij} and $-a_{ij}$, respectively. Note that the sum of the two payoffs is zero, which explains the name of the game. If two players employ mixed strategies *x* and *y* (and a pure strategy is just a special case of a mixed strategy), then the payoff to player 1 is:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_i y_j a_{ij}$$

which can be seen as the expected payoff among all of the pure strategies represented by *x* and *y*.

Therefore, a TPZS game is completely defined when the payoff for each pair of *X* and *Y* pure strategies is determined. These payoffs can be summarized in an $m \times n$ matrix, generally referred to as a *payoff matrix*, i.e.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

and the payoff to player 1 is then $x^T Ay$.

In playing the game, both players are assumed to choose a strategy that achieves the most favorable outcome. This means that *X* would choose the strategy that maximizes $x^T Ay$ over all choices of *y*. On the other hand, *Y* would choose the strategy that minimizes $x^T Ay$ over all choices of *x*.

A TPZS game has an *equilibrium point* when each player can guarantee an optimal result by always choosing a single pure strategy. When equilibrium cannot be achieved, players must use mixed strategies to optimize the value of the game.

To choose the best randomized strategy, *X* must find the $x = (x_1, \ldots, x_m)$ to

$$\max_x \{\min_j [\sum_{i=1}^{m} x_i a_{ij}]: \sum_{i=1}^{m} x_i = 1,\ x_i \geq 0,\ i = 1,...,m\}.$$

Similarly, *Y* must find the $y = (y_1, \ldots, y_n)$ to

$$\min_y \{\max_i [\sum_{j=1}^{n} a_{ij} y_j]: \sum_{j=1}^{n} y_j = 1,\ y_j \geq 0,\ j = 1,...,n\}.$$

Let x^* and y^* denote the optimal strategies for *X* and *Y*. Then, $v^* = (x^*)^T Ay^*$ is the *value* of the game. One of the central results of game theory states (Winston, 1991) that:

$$v^* = \max_x \{ \min_j [\sum_{i=1}^{m} x_i a_{ij}]: \sum_{i=1}^{m} x_i = 1,\ x_i \geq 0,\ i = 1,...,m \}, \text{ and}$$

$$v^* = \min_y \{ \max_i [\sum_{j=1}^{n} a_{ij} y_j]: \sum_{j=1}^{n} y_j = 1,\ y_j \geq 0,\ j = 1,...,n \}.$$

B. LINEAR PROGRAMMING

When the payoff matrix is specified and it is not too large, linear programming (Winston, 1991) can be used to find the optimal mixed strategies and the value of the game. For the maximizer (player 1), the problem is to find the mixed strategy $x = (x_1, \dots, x_m)$ which maximizes $\min_j \sum_{i=1}^{n} x_i a_{ij}$. That is,

$$\begin{aligned} \text{LP1:} \quad & \max \quad v \\ \text{subject to} \quad & \sum_{i=1}^{m} x_i a_{ij} - v \geq 0, \quad j = 1,...,n \\ & \sum_{i=1}^{m} x_i = 1, \quad \text{and} \quad x_i = 1,...,m. \end{aligned}$$

Similarly, the minimizer (player 2) must solve LP 2:

$$\begin{aligned} \text{LP2:} \quad & \min \quad w \\ \text{subject to} \quad & \sum_{j=1}^{n} y_j a_{ij} - w \leq 0, \quad i = 1,...,m \\ & \sum_{j=1}^{n} y_j = 1, \quad \text{and} \quad y_j = 1,...,n. \end{aligned}$$

It is easy to show that problems LP1 and LP2 are duals of each other. Moreover, if (v^*, x^*) and (w^*, y^*) are optimal to problems LP1 and LP2, respectively, then $v^*=w^*$.

C. FICTITIOUS PLAY (FP): BROWN-ROBINSON METHOD

Fictitious play (FP) was introduced by Brown and Robinson (1951). It is an iterative solution procedure; in each iteration, players choose pure strategies that are the best response to the empirical mix of their opponents' pure strategies seen so far.

The FP procedure implemented here begins at iteration 1 with player 1 selecting that row maximizing the minimum row value, and player 2 selecting that column minimizing the maximum column value. Denote the players' pure strategies at iteration 1 as $x^{(1)}$ and $y^{(1)}$. These are vectors of all zeros except for a 1 at the selected row or column locations. At iteration 2, player 1 selects pure strategy $x^{(2)}$, which is the best row response to $y^{(1)}$; and player 2 selects pure strategy $y^{(2)}$, which is the best column response to $x^{(1)}$. And for general iteration $k \geq 2$, player 1 selects the pure strategy $x^{(k)}$, which is the best row response to

$$\overline{y}^{(k-1)} = \frac{1}{k-1}\sum_{p=1}^{k-1} y^{(p)} \text{ , and}$$

player 2 selects pure strategy $y^{(k)}$, which is the best column response to

$$\overline{x}^{(k-1)} = \frac{1}{k-1}\sum_{p=1}^{k-1} x^{(p)} \text{ .}$$

For computational purposes, $\overline{x}^{(k+1)}$ is conveniently updated from $\overline{x}^{(k)}$ and $x^{(k+1)}$ as follows:

$$\overline{x}^{(k+1)} = \left(\frac{k}{k+1}\right)\cdot \overline{x}^{(k)} + \left(\frac{1}{k+1}\right)\cdot x^{(k+1)} \text{ .}$$

And similarly for player 2,

$$\overline{y}^{(k+1)} = \left(\frac{k}{k+1}\right)\cdot \overline{y}^{(k)} + \left(\frac{1}{k+1}\right)\cdot y^{(k+1)} \text{ .}$$

Any limit points of the sequences $\{\overline{x}^{(k)}\}$ and $\{\overline{y}^{(k)}\}$ are optimal mixed strategy solutions to the game. Also upper and lower bounds on the value of the game, v^*, are determined at each game play. Specifically, at game iteration k,

$$\underline{v}_k \equiv (\overline{x}^{(k-1)})^t A y^{(k)} \leq v^* \leq (x^{(k)})^t A \overline{y}^{(k-1)} \equiv \overline{v}_k \text{ ,}$$

and both $\underline{v}_k$ and $\overline{v}_k$ converge to v^*, but not necessarily monotonically (Eagle and Washburn, 1991).

D. BLOTTO GAMES

1. Definition

In a Blotto game, there are $n \geq 1$ targets, or *areas*, for player 1 (the attacker) to attack and player 2 (the defender) to defend. The attacker chooses a vector of allocations x, where x_k is the number of attacking units assigned to area k, and player 1 has f attackers to distribute, resulting in the constraint $\sum_{k=1}^{n} x_k \leq f$. The defender chooses a vector of allocations y subject to $\sum_{k=1}^{n} y_k \leq g$, where y_k is the number defenders assigned by player 2 to area k. The payoff is $\sum_{k=1}^{n} A(x_k, y_k)$ (See Washburn, 1994, pp.107-111 for a more complete discussion). All allocations are required to be nonnegative, and in a discrete Blotto game they are also required to be integers. The number of pure strategies for player 1 is $\binom{n+f-1}{f}$ and, for player 2, $\binom{n+g-1}{g}$, both of which grow too fast to allow complete enumeration in even moderately sized games. Figure 1 displays a typical increase in the number of pure strategies for player 1 as f or n increase.

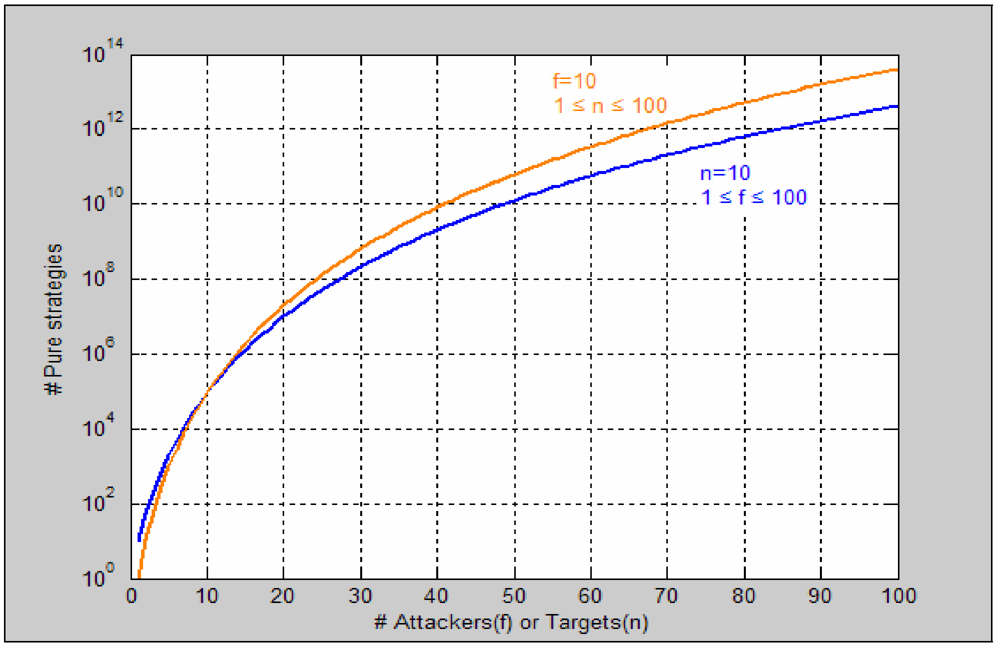

Figure 1. Number of pure strategies for player 1, for f=10 $1 \leq n \leq 100$, and for n=10,

$$1 \leq f \leq 100.$$

2. Playability

In Blotto games, it is sometimes convenient to represent a mixed strategy with the marginal distributions of the random vector $X=(X_1,\ldots, X_n)$, where X_i is the random variable representing the number of attackers assigned to area i. Marginal distributions satisfying $\sum_k X_k = f$ are playable for the attacker. Similarly, the marginals for the random variable $Y= (Y_1,\ldots, Y_n)$ are playable for the defender if $\sum_k Y_k = g$. However, the typical approach is to relax these restrictions and simply require that $\sum_k E(X_k) = f$ and $\sum_k E(Y_k) = g$.

3. ILP Formulation of Blotto Games

Washburn (1994) presents the LP formulation of Blotto games using the marginal distributions. However, those formulations do not guarantee playability (although the discussions of those formulations claim that playability is not an issue for large problems). We modify the formulation in Washburn (1994) by explicitly adding constraints that are sufficient to enforce playability. The cost of such constraints is twofold: (1) they are sufficient, but not necessary conditions for playability, so the solutions we obtain are potentially suboptimal; and (2) they introduce integer variables, rendering integer linear programming (ILP) formulations.

ILP 1 solves a Blotto defense game for the defender's marginals $(y_1,\ldots,y_g)$, where y_i represents the probability that j defenders are used in any given area. Therefore, $\sum_k j \cdot y_i = E(Y_k)$.

$$
\begin{aligned}
\text{ILP1:} \quad & \min_{\text{y, c, d}} \quad v = nc + df \\
& \text{s.t} \quad \sum_{j=0}^{g} A(i,j)\cdot y_j \le c + d\cdot i; \quad i = 0,1\ldots, f \\
& \sum_{j=0}^{g} y_j \cdot j \le g/n \\
& \sum_{j=0}^{g} y_j = 1 \\
& ycount_j = n \cdot y_j \\
& y_j \ge 0, \quad \text{for all } j = 0,1,\ldots, g \\
& ycount_j \in \{0,\ldots, g\} \ \text{ and } \ d \ge 0
\end{aligned}
$$

ILP 2 solves a Blotto attack game for the attacker's marginals $(x_1,\ldots,x_f)$. The integer restrictions or $ycount_j$ and $xcount_i$ are used to require playability.

$$
\begin{aligned}
\text{ILP2:} \quad & \max_{\text{x, a, b}} \quad v = na - bg \\
& \text{s.t} \quad \sum_{i=1}^{f} A(i,j)\cdot x_i \ge a - b\cdot j; \quad j = 0,1\ldots, g \\
& \sum_{i=1}^{f} x_i \cdot i \le f/n \\
& \sum_{i=1}^{f} x_i = 1 \\
& xcount_i = n \cdot x_i \\
& x_i \ge 0, \quad \text{for all } i = 0,1,\ldots, f \\
& xcount_i \in \{0,\ldots, f\} \ \text{and} \ b \ge 0
\end{aligned}
$$

These playability constraints are too restrictive; they are sufficient to enforce playability but they are provably not necessary.

4. New Fictitious Play Procedure

We derive the dynamic programming recurrence relation for solving Blotto games with FP, using a general payoff function $A_k(x, y)$, which is the amount player 2 pays to player 1 when player 1 allocates *x* units to area *k* and player 2 allocates *y* units to area *k*. The total payoff is obtained by summing the rewards over the *n* areas. The recurrence can be solved without explicitly keeping track of every attack or defense played; rather, the information required is simply the number of times a given force level (number of attackers or defenders) has been used in each area, over all attacks and defenses seen so far.

We first consider the defender's problem at FP iteration *K*, which is to allocate *g* defenders over *n* cities to minimize the expected payoff, given that *K* attacks have been observed so far. Each attack can be represented by a column vector $a^k = (a_1^k, a_2^k, \ldots, a_n^k)^T$, $k = 1, \ldots, K$. We define the values $r_i^j = \sum_k I_{(a_i^k = j)}$, where $I_{(a_i^k = j)}$ represents the indicator variable for the event, "the kth attack used *j* attackers in area *i*." Therefore, r_i^j represents the number of times exactly *j* attackers have been used against area *i*. We first determine the value of placing *g* defenders optimally in area *n*. Then we define a recurrence relation on a value function $v_i(p)$ that represents the expected payoff of placing *p* defenders optimally in areas *i*, *i*+1,…, *n*. Then $v_1(g)$ is the solution to the original problem. The boundary condition is given by

$$v_n(q) = \frac{1}{K} \sum_{p=0}^{f} r_n^p A_n(p, q), \ q = 0, \ldots, g \ , \tag{1}$$

which is the total expected payoff when the defender uses *q* defenders in area *n*. This states that the optimal defender strategy when only area *n* remains is to allocate all *q* remaining defenders to that area. The recurrence for $i \in \{1, \ldots, n-1\}$ is:

$$v_i(q) = \frac{1}{K} \min_{j=0,\ldots,q} \left\{ \sum_{p=0}^{f} r_i^p \ A_i(p, j) + v_{i+1}(q - j) \right\}. \tag{2}$$

This is the expected payoff in area *i* plus the expected payoff generated by placing the remaining (*q*-*j*) defenders optimally in areas *i*+1,…, *n*. The optimal defender allocation to area *i* is the value of *j* minimizing equation (2).

Similarly, for the attacker's problem, we assume that *K* defenses have been observed so far, where the *k*th defense is $d^k = (d_1^k, d_2^k, \ldots, d_n^k)^T$. The attacker wishes to allocate *f* forces over the *n* areas to maximize the expected payoff. Let $s_i^j = \sum_k I_{(d_i^k = j)}$ be the number of times *j* defenders are placed in area *i*. Define $w_i(p)$ as the maximum possible expected payoff in areas $i, \ldots, n$. The boundary conditions are:

$$w_n(p) = \frac{1}{K} \sum_{q=0}^{g} s_n^q \, A_n(p,q), \quad p = 0, \ldots, f \ , \tag{3}$$

and the recurrence is given by:

$$w_i(p) = \frac{1}{K} \max_{j=0,\ldots,p} \left\{ \sum_{q=0}^{g} s_i^q \, A_i(j,q) + w_{i+1}(p-j) \right\}. \tag{4}$$

The optimal solution for the attacker is represented by $w_1(f)$ and the corresponding decisions *j* maximizing (4) for each area.

Blotto games can be extended immediately to the case where the attacker possesses different numbers of, say, two *types* of attacking units, f_1 and f_2, and the defender also has a supply of, say, two types of defending unit, g_1 and g_2. The payoff function now depends on the number of attackers and defenders of each type allocated to each area: $A_i(p_1, p_2, q_1, q_2)$. If we define $r_i^{j_1, j_2}$ as the number of times j_1 attackers of type 1 and j_2 of type 2 have been used in area *i*, and similarly for $s_i^{j_1, j_2}$, then our value functions are two-dimensional, with boundary conditions

$$v_n(q_1, q_2) = \frac{1}{K} \sum_{p_1=0}^{f_1} \sum_{p_2=0}^{f_2} r_n^{p_1, p_2} A_n(p_1, p_2, q_1, q_2), \tag{5}$$

and

$$w_n(p_1, p_2) = \frac{1}{K} \sum_{q_1=0}^{g_1} \sum_{q_2=0}^{g_2} s_n^{q_1,q_2} A_n(p_1, p_2, q_1, q_2)\,, \tag{6}$$

and recurrences

$$v_i(q_1, q_2) = \frac{1}{K} \min_{\substack{j_1=0,\ldots,q_1 \\ j_2=0,\ldots,q_2}} \left\{ \sum_{p_1=0}^{f_1} \sum_{p_2=0}^{f_2} r_i^{p_1,p_2} A_i(p_1, p_2, j_1, j_2) + v_{i+1}\left(q_1 - j_1, q_2 - j_2\right) \right\}. \tag{7}$$

and

$$w_i(p_1, p_2) = \frac{1}{K} \max_{\substack{j_1=0,\ldots,p_1 \\ j_2=0,\ldots,p_2}} \left\{ \sum_{q_1=0}^{g_1} \sum_{q_2=0}^{g_2} s_i^{q_1,q_2} A_i(j_1, j_2, q_1, q_2) + w_{i+1}(p_1 - j_1, p_2 - j_2) \right\}. \tag{8}$$

Clearly, the size of the static space grows with the product of the number of each type of attacker or defender. It is still manageable with just a few types of attacker or defender.

III. DATA ANALYSIS AND RESULTS

A. MODEL IMPLEMENATION

The FP model is implemented in MATLAB (Version 6.5). The ILP solution procedure is implemented in GAMS (Revision 135, XA solver). Computations are done on a 1.5 GHz Intel Centrino-based laptop computer with 512 MB of RAM. All computer code appears in Appendices A and B.

B. NUMERICAL RESULTS

1. Rate of Convergence of FP

Define *gap*(*k*) to be the difference between the upper and lower bounds on the value of the game at FP iteration *k*. Consistent with earlier FP studies (Washburn, 2001), we find that the FP gap plotted against number of iterations is approximately asymptotically linear on a log-log plot. That is, for large enough *k*,

$$gap(k) \approx a/k^{b} \text{ , or}$$

$$\log(gap(k)) \approx \log(a) - b\log k \text{ ,}$$

where k is number of iterations and a and b are fitted constants. Limited numerical experimentation suggests that using a least square fit and dropping first 100 iterations the intercept (log(*a*)) increases with increasing *f* or *g*, and the slope (-*b*) increases (to approximately -1/2) with increasing *n*. These observations are illustrated in Figures 2, 3 and 4.

Case	*f*, *g*, *n*	# pure strategies		Slope	Intercept	Final Gap (UB-LB)
		Attacker	Defender			
A1	50, 200, 10	12565671261	1.76081E+15	-0.48374	1.8887	3.8396
A2	25, 100, 10	52451256	4.26342E+12	-0.48378	1.5860	1.5260
A3	5, 20, 10	2002	10015005	-0.48376	0.8887	0.3052

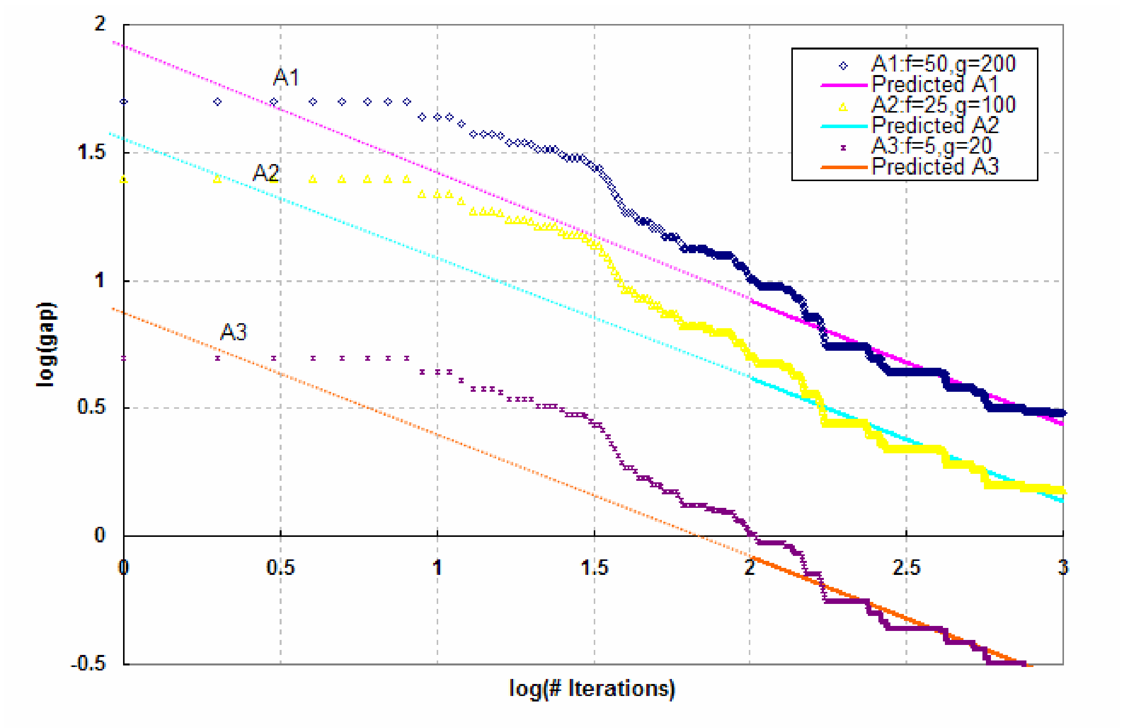

Figure 2. With *n* fixed, the slope remains constant and the intercept increases with *f* and *g*.

In Figure 2 we see the gap between the upper and lower bounds plotted against the number of iterations of fictitious play, on log-log scale. The slope of the fitted line (from the column labeled "slope" in the table above the plot) indicates the rate of convergence. Note that the slope is constant as *f* and *g* increase, and *n* remains fixed.

Case	$f,\ g,\ n$	# pure strategies		Slope (a)	Intercept (b)	Final Gap (UB-LB)	Standard Error
		Attacker	Defender				
B1	30, 50, 30	5.91E+16	3.33E+21	-0.517	1.8541	2.0374	0.0094
B2	30, 50, 20	1.89E+13	4.63E+16	-0.543	1.8954	1.7824	0.0157
B3	30, 50, 15	1.15E+11	4.79E+13	-0.580	1.9463	1.5238	0.0187
B4	30, 50, 10	2.12E+08	1.26E+10	-0.588	1.8839	1.1999	0.0270
B5	30, 50, 5	46376	316251	-0.851	2.1746	0.4787	0.0469
B6	30, 50, 2	31	51	-1.093	1.4431	0.0216	0.0194

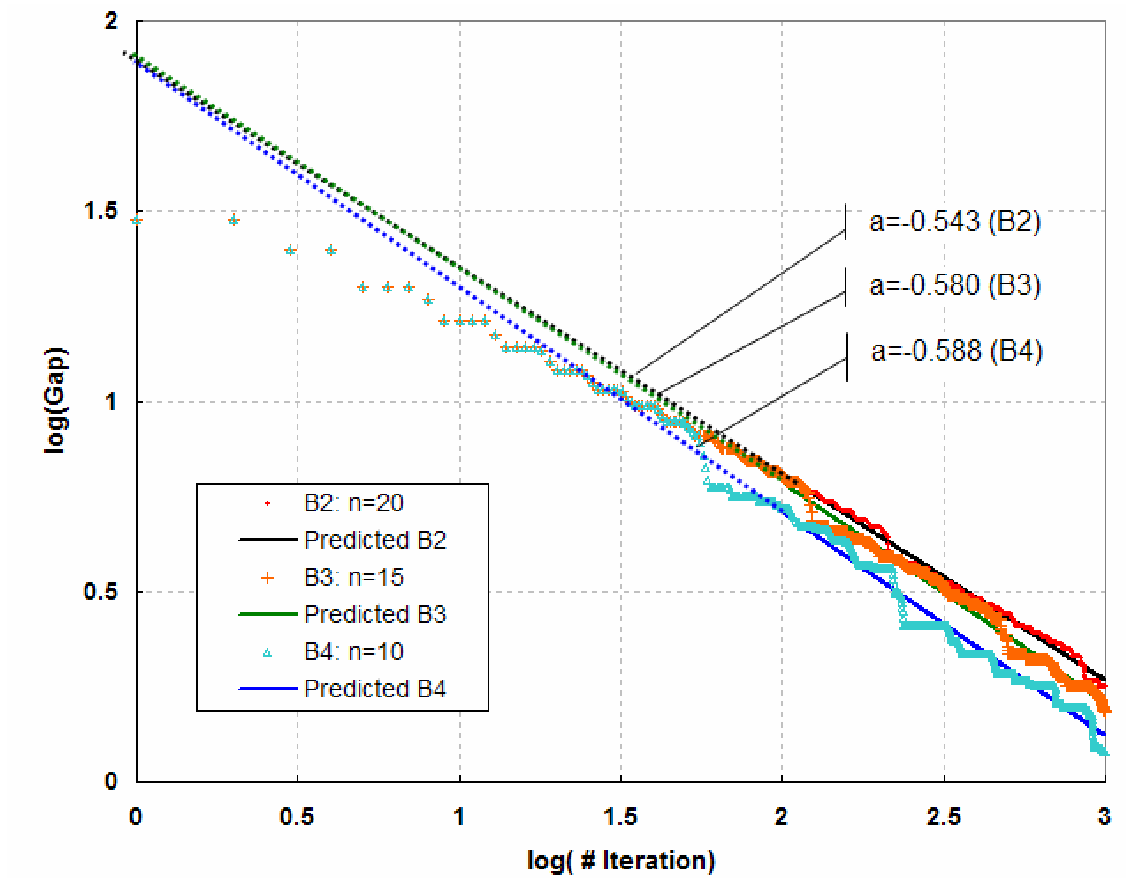

Figure 3. The best-fit slope increases with *n* (*f* and *g* fixed).

If we increase n for a fixed f and g, we see in Figure 3 that the slope increases, and appears to approach a limit of -0.5 (Figure 4). This is consistent with conjectures of $1/\sqrt{k}$ convergence of FP.

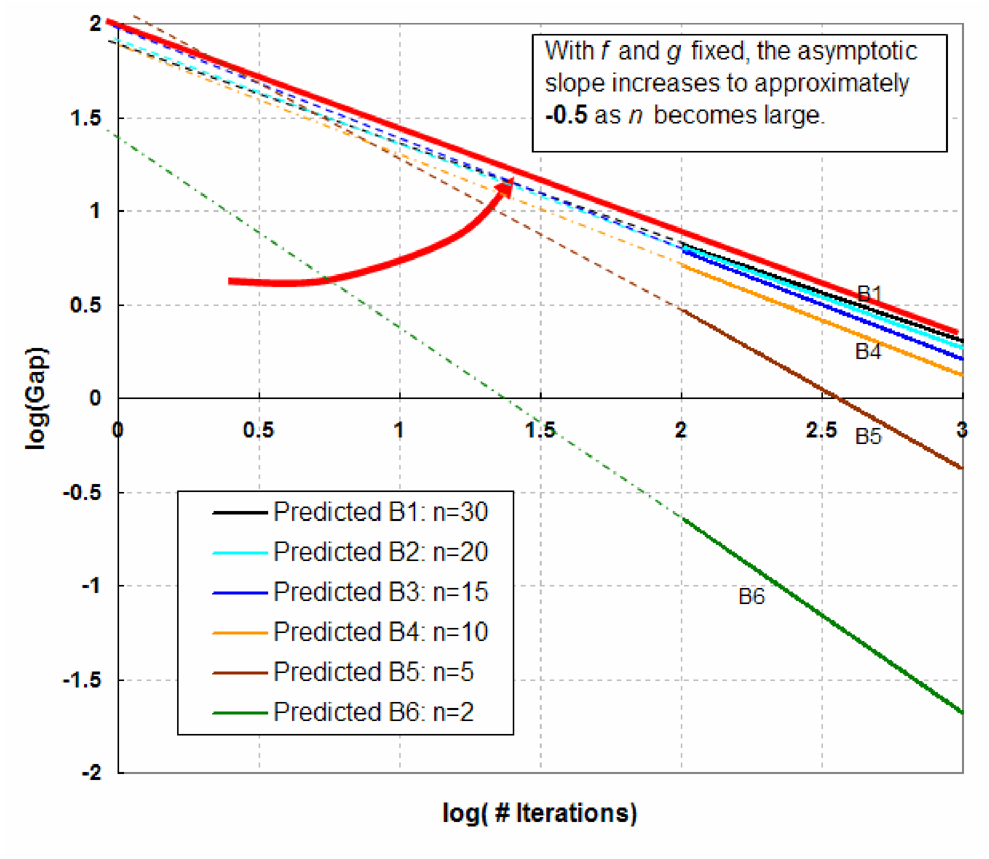

Figure 4. Convergence of asymptotic slope

2. Elapsed Time per FP Iteration

We observe that for all tested values of *f*, *g* and *n*, the elapsed time per FP iteration is constant as the number of FP iterations *k* increases. This is illustrated in Figure 5 and occurs because the amount of FP data required to be manipulated and stored does not increase with *k*.

Iteration	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
f, *g*, *n*	Elapsed time (sec)									
5, 6, 10	2.4	4.6	6.8	9.0	11.2	13.3	15.5	17.7	19.9	22.2
20, 25, 30	24.3	48.4	72.6	96.7	121.4	145.5	169.1	193.8	217.6	242.0
40, 45, 50	81.7	163.4	245.2	326.4	409.2	490.4	572.2	653.1	736.6	817.4

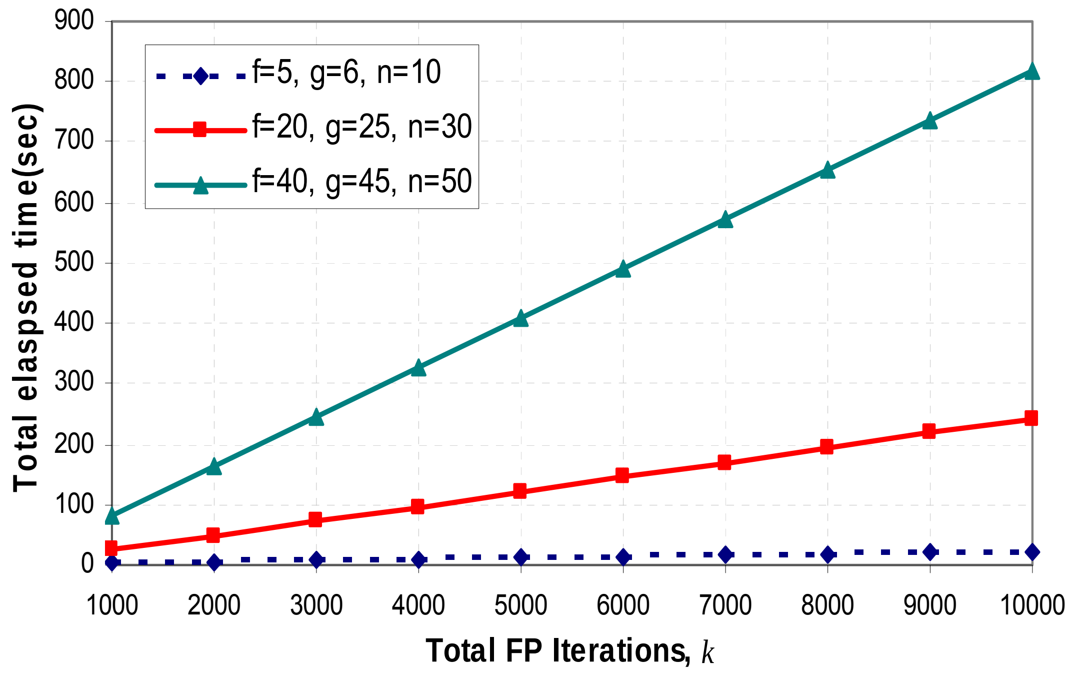

Figure 5. FP iterations vs. Elapsed time for 3 games

3. Comparisons with FP and ILP Procedure

Comparisons are made between the ILP and FP solution procedures. Three different payoff functions are examined.

a. Convex Payoff Function

The convex payoff function is given by

$$A(x_i, y_i) = \max(x_i - y_i, 0).$$

Figure 6 shows how the two procedures performed.

Run	$f,\ g,\ n$	FP ($k = 2000$)				ILP	
		Elapsed time	Upper	Lower	Gap	Elapsed time	Value of game
1	3, 4, 5	1.962	2.208	2.179	0.029	0.10	2.2
2	6, 8, 5	2.774	4.415	4.358	0.057	0.18	4.4
3	12, 16, 5	4.657	8.831	8.717	0.114	0.22	8.8
4	15, 20, 5	5.668	11.039	10.896	0.143	0.19	11
5	30, 40, 5	11.147	22.077	21.792	0.285	0.25	22
6	60, 80, 5	24.535	44.154	43.584	0.570	0.12	44
7	120, 160, 5	59.766	88.308	87.168	1.140	0.21	88
8	150, 200, 5	83.801	110.385	108.960	1.426	0.12	111

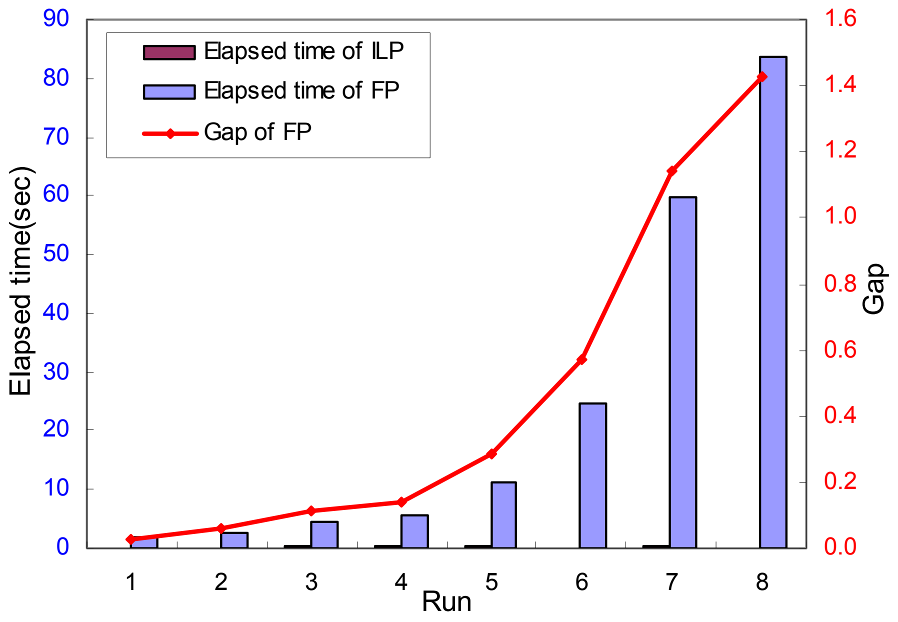

Figure 6. Comparison of FP and ILP procedures with convex payoff function

b. Capacitated Payoff Function

The capacitated payoff function is

$$A(x_i, y_i) = \begin{cases} \max(x_i - y_i, 0), & x_i - y_i \le cap \\ cap, & x_i - y_i > cap \end{cases},$$

where *cap* is the maximum possible payoff. We note that the ILP objective function need not be either convex or concave. Figure 7 shows how the two procedures performed.

Run	*f, g, n*	FP (k = 2000)				ILP			
		Elapsed time	Upper	Lower	Gap	Elapsed time	Upper	Lower	Gap
1	3, 4, 5	1.743	2.208	2.179	0.029	0.180	2.200	2.200	0.000
2	6, 8, 5	2.704	2.958	2.872	0.086	0.100	3.120	2.800	0.320
3	12, 16, 5	4.697	3.935	3.767	0.168	0.100	4.114	3.800	0.314
4	15, 20, 5	5.748	4.252	4.078	0.174	0.170	4.333	4.000	0.333
5	30, 40, 5	11.226	4.884	4.619	0.265	0.190	5.143	4.200	0.943
6	60, 80, 5	24.536	5.350	4.960	0.389	1.420	5.807	4.200	1.607
7	120, 160, 5	59.946	5.622	5.130	0.492	62.380	6.250	4.200	2.050
8	150, 200, 5	82.278	5.713	5.144	0.570	102.170	6.338	4.091	2.247

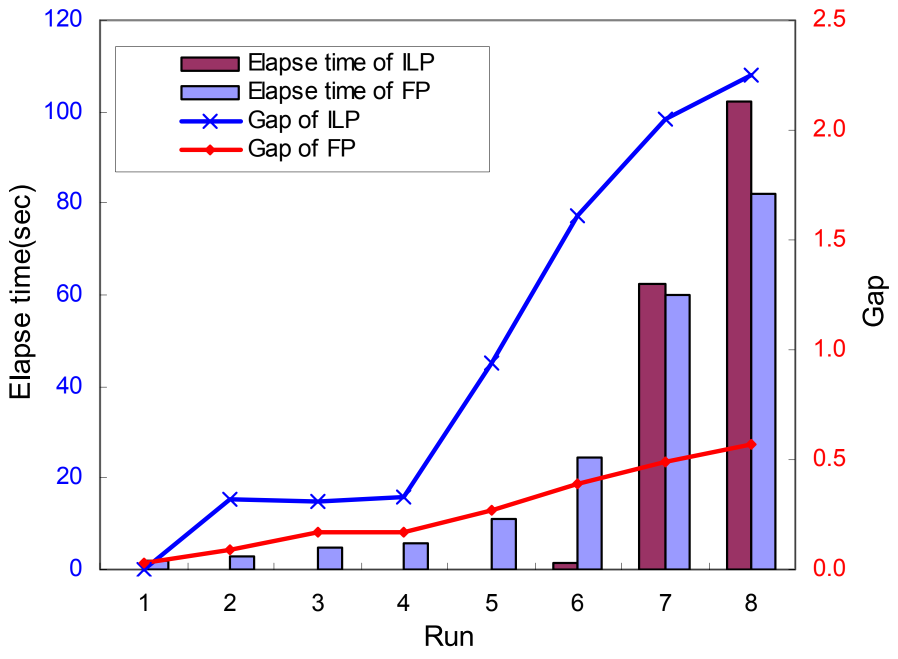

Figure 7. Comparison of FP of ILP procedures with the capacitated payoff function

c. Binary Payoff Function

The binary payoff function is

$$A(x_i, y_i) = \begin{cases} 0, & x_i - y_i \le 0 \\ 1, & x_i - y_i > 0 \end{cases} .$$

As with the capacitated payoff function, the ILP in this case need not be either convex or concave. Figure 8 shows how the two procedures performed.

Run	*f*, *g*, *n*	FP ($k = 2000$)				ILP			
		Elapsed time	Upper	Lower	Gap	Elapsed time	Upper	Lower	Gap
1	3, 4, 5	1.021	1.216	1.179	0.037	0.12	1.200	1.200	0.000
2	6, 8, 5	2.805	1.477	1.407	0.070	0.17	1.600	1.400	0.200
3	12, 16, 5	4.757	1.675	1.574	0.101	0.19	1.920	1.400	0.520
4	15, 20, 5	5.668	1.721	1.618	0.103	0.11	2.000	1.333	0.667
5	30, 40, 5	11.186	1.850	1.694	0.156	0.30	2.000	1.400	0.600
6	60, 80, 5	24.756	1.921	1.716	0.205	2.57	2.167	1.400	0.767
7	120, 160, 5	60.758	1.990	1.703	0.287	61.96	2.182	1.400	0.782
8	150, 200, 5	83.260	2.012	1.686	0.326	149.89	2.214	1.400	0.814

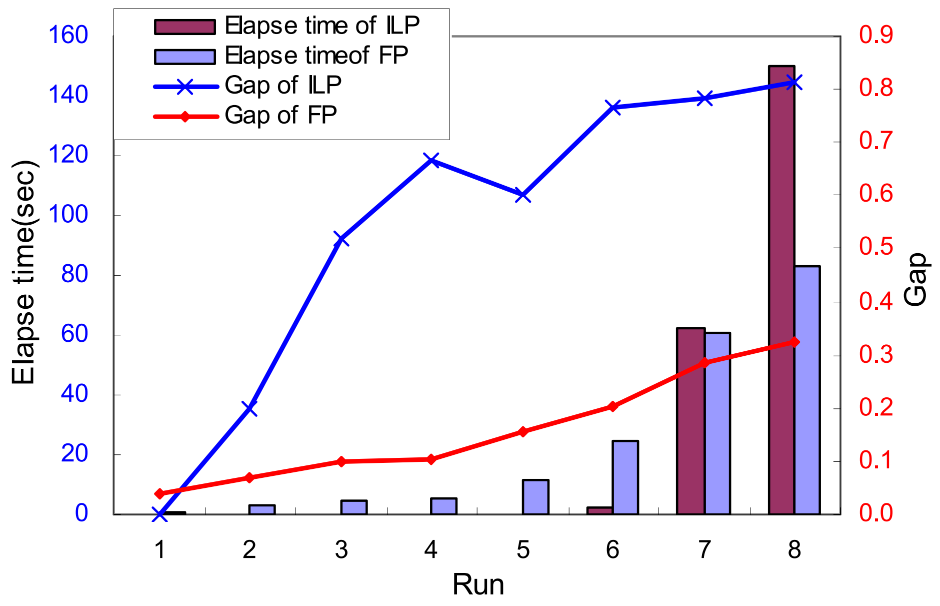

Figure 8. Comparison of FP and ILP procedures with the binary payoff function

C. CONCLUSIONS

As has been observed in earlier FP studies the FP gap (the difference between the upper and lower bounds or game value) as a function of number of FP iterations, *k*, is approximately

$$gap(k) \approx {}^{a}\!/_{k^{b}} ,$$

for large enough k.

The best-fit *a* increases with *f* and *g*, and the best-fit *b* decreases to approximately 1/2 with increasing *n*.

Because of efficiencies realized in the DP procedure used to solve the FP subproblems, the computation time required for each FP iteration is approximately constant as the number of FP iterations increases, for fixed *f*, *g* and *n*.

For the convex payoff function tested, the ILP formulation solved with GAMS was faster and more accurate than the FP procedure.

For the non-convex payoff functions tested, the FP procedure was more competitive and sometimes significantly outperformed than ILP procedure.

IV. CONCLUSIONS AND FURTHER STUDY

We propose a new efficient fictitious play (FP) procedure to solve two-person zero-sum Blotto games. The algorithm uses dynamic programming (DP) to solve the FP subproblems at each iteration. By representing intermediate mixed strategies through marginal distributions are keep the state space of the DP manageable and independent of the number of iterations. Although our experiments considered one type of attacker and one type of defender, we indicate how to generalize this procedure to cases with more than one type of attacker or defender (or both).

During this study, we identified other topics for further investigations. The first is to investigate generalizations of Blotto games in which defenders can be placed in such a way as to defend multiple areas at once. This is closer to the real situation with missile defense. The second is to explore the issue of playability in the ILP formulations. Our proposed playability constraint is currently too restrictive; we have provided examples in which the optimal solution to the ILP, with the playability constraint, is not equal to the value of the game. Further research should explore less restrictive, alternate formulations of playability constraints. It is possible (although unlikely) that less restrictive playability constraints would also yield more efficiently solvable ILP, making that approach competitive with the DP-based procedure for larger problems.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB CODE FOR NEW FP PROCEDURE

1. Blotto_fp.m

```
% Blotto Game Fictitious Play Solution with general payoff matrix

% n = # areas to be attacked and defended
% f = # attackers to be allocated to the n areas
% g = # defenders to be allocated to the n areas

%  Initialize with a vector of marginals for the attacker and
%  defender.
%  initialize A for attacker (each row is frequency of
%  0,1,2. ... ,f being assigned to area i=1,...,n),
%  initialize D for defender (each row is frequency of
%  0,1,2, ... ,g being assigned to area i=1,...,n),
%  initalize upper and lower bounds on the value of the game, and
%  number of iterations desired.
%  Calculate P(a,d) = return if a attackers and d defenders
%  attack any area

clear;

f = 150;          % number of attackers
g = 200;          % number of defenders
n = 5;            % number of areas to attack and defend
num_its = 2000;   % number of FP iterations

for i = 1:f+1
      for j= 1:g+1
            P(i,j)=linearpayoff(i,j);
%       P(i,j)=cappedpayoff(i,j);
%       P(i,j)=binarypayoff(i,j);

      end
end

% Set initial A for all attacks in area 1
A(1,:) = [zeros(1,f),1];
for k=2:n
      A(k,:) = [1,zeros(1,f)];
end

% Set initial D to all defenses in area 1
D(1,:) = [zeros(1,g),1];
for k=2:n
      D(k,:) = [1,zeros(1,g)];
end

v_up = f;  % assumes all missiles are leakers
v_low = 0; % assumes no missiles are leakers
       % preallocates v_up and v_low in memory
v_up=v_up*ones(1,num_its+1);
```

```
      v_low=v_low*ones(1,num_its+1);

      for k=1:num_its
            [opt_atk,v_u] = attacker(D,P,n,f,g);
      % Find best pure attack for defenses seen so far
      % (and upper bound).
            [opt_def,v_l] = defender(A,P,n,f,g);
      % Find best pure defense for attacks seen so far
      % (and lower bound).
            A = update_attack_marginals(A,opt_atk);
      %Update attack and defense marginals.
            D = update_defense_marginals(D,opt_def);
            v_up(k+1) = min(v_u,v_up(k));
            v_low(k+1)= max(v_l,v_low(k));
            if  (k/100  ==  floor(k/100)),  home,  k,  gap=(v_up(k+1)-
v_low(k+1)), end
      end

      figure(1)
      loglog([0:num_its],(v_up  -  v_low),'ro'),grid  on,  title('(upper
bound - lower bound) vs. #FP iterations')
      figure(2)
      plot([0:num_its],v_up,'go',[0:num_its],v_low,'rx'),grid        on,
axis([0  num_its  0  f]),title('Upper  and  lower  bounds  vs.  #FP
iterations')

      bounds = [v_up(end),v_low(end)]
      meanA = mean(A)/num_its
      meanD = mean(D)/num_its
```

2. Attacker.m

```
function [opt_attack,v_up] = attacker(D,P,n,f,g)

% Takes defenses D and returns optimal attack column vector and
% an upper bound on the value of the Blotto game.
% n areas, f attackers, g defenders
% Uses general payoff function P(a,d)
% #defenses (increases with FP iterations)

num_defenses = sum(D(1,:));

s=D*P';          % compute s(i,j) = exp. 1-step payoff
% when j attackers assigned to area i.
% Uses the marginal matrix
% D(i,j) = # times j defenders assigned to area i.

v_star = zeros(n,f+1);  %initialize v_star(i,j) = optimal
% return when j attackers are available for i areas
a_star = zeros(n,f+1); %initialize a_star(i,j) = optimal
% # attackers to use when j attackers are available for i areas

% n = 1
v_star(1,:) = s(1,:); % optimal payoff when 1 area is included
a_star(1,:) = [0:f]; % optimal #attackers when 1 area is included

% n > 1
for i=2:n          % areas
      for j=0:f     % j attackers remain to be used
            v_starnew = zeros(1,j+1);  %initialize a_starnew
            for k=0:j
                  v_starnew(k+1) = s(i,k+1) + v_star(i-1,j-k+1);
% enumerating possible returns
            end
            [v_star(i,j+1),a_star(i,j+1)] = max(v_starnew);
% identifying the # attackers giving a max. payoff
            a_star(i,j+1) = a_star(i,j+1)-1;
% correcting for col. 1 being for 0 attackers
      end
end

v_up = v_star(n,f+1)/num_defenses;
opt_attack = zeros(n,1);   % initialize opt_attack
opt_attack(n) = a_star(n,f+1);
% establish optmal #attackers to use for n areas

attackers_remaining = f - opt_attack(n);
% update #attackers remaining for remaining n-1 areas

for i=n-1:-1:1
      opt_attack(i) = a_star(i,attackers_remaining+1);
      attackers_remaining = attackers_remaining - opt_attack(i);
end
```

3. Defender.m

```
function [opt_defense,v_low] = defender(A,P,n,f,g)

% Takes attacks A and returns optimal defense column vector and
% an lower bound on the value of the Blotto game.
% n areas, f attackers, g defenders
% Uses general payoff function P(a,d)

num_attacks = sum(A(1,:));

q=A*P;      % compute q(i,j) = exp. 1-step payoff
% when j attackers assigned to area i.
                % Uses the marginal matrix
% A(i,j) = # times j attackers assigned to area i.

r_star = zeros(n,g+1);    %initialize r_star and d_star
d_star = zeros(n,g+1);

% n = 1
r_star(1,:) = q(1,:);
d_star(1,:) = [0:g];

% n > 1
for i=2:n
      for j=0:g
            r_starnew = zeros(1,j+1);  %initialize r_starnew
            for k=0:j
                  r_starnew(k+1) = q(i,k+1) + r_star(i-1,j-k+1);
% enumerating possible # defenders
            end
            [r_star(i,j+1),d_star(i,j+1)] = min(r_starnew);
% identifying the # defenders giving a min. payoff
            d_star(i,j+1) = d_star(i,j+1)-1;
% correcting for col. 1 being for 0 defenders
      end
end

v_low = r_star(n,g+1)/num_attacks;
opt_defense = zeros(n,1);
opt_defense(n) = d_star(n,g+1);
defenders_remaining = g - opt_defense(n);
for i=n-1:-1:1
      opt_defense(i) = d_star(i,defenders_remaining+1);
      defenders_remaining = defenders_remaining - opt_defense(i);
end
```

4. Update_attack_marginals.m

```
function [A_new] = update_attack_marginals(A,opt_atk)
[n,m]=size(A);
A_new=A;
for k=1:n
      z=opt_atk(k);
      A_new(k,z+1)=A_new(k,z+1)+1;
End
```

5. Update_defense_marginals.m

```
function [D_new] = update_defense_marginals(D,opt_def)
[n,m]=size(D);
D_new=D;
for k=1:n
      z=opt_def(k);
      D_new(k,z+1)=D_new(k,z+1)+1;
end
```

6. Payoff.m

Unconstrained target case:

Linearpayoff.m
```
function P=linearpayoff(a,d)
if a>d
      P=a-d;
else
      P=0;
end
```

Capacitated target case:

Cappedpayoff.m
```
function P = cappedpayoff(a,d,cap)
P=max(0,min(a-d,cap));
```

Binary target case:

Binarypayoff.m
```
function P = binarypayoff(a,d)
P=(a>d);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. GAMS CODE FOR ILP PROCEDURE

```
*  Blotto Attack & Defense game
*  Attacker is the row player and maximizer.
*  Defender is the column player and minimizer.
*  Uses integer restrictions to require playability

$offlisting
$inlinecom { }
OPTIONS
         SOLPRINT =     OFF,
         DECIMALS =       2,
         LIMCOL   =       0,
         LIMROW   =       0,
         RESLIM   =   86400,   {MAX SECONDS}
         ITERLIM  =   100000,  {MAX PIVOTS}
         OPTCA    =     0.01,  {ABSOLUTE INTEGRALITY TOLERANCE}
         OPTCR    =     0.00,  {RELATIVE INTEGRALITY TOLERANCE}
         lp       =    xa,
         MIP      =    xa; {XA Solver}

SCALARS
         N       total number of areas       /5  /
         F       total number of attackers   /15 /
         G       total number of defenders   /20 /
         Time    execution time              /0  /
;

SETS
         i       # attackers used   /0attack*15attack/
         j       # defenders used   /0defend*20defend/
;

PARAMETER P(i,j) damage done when i attack and j defend;

LOOP ((i,j), P(i,j)=max((ord(i)-ord(j)),0)); {convex payoff}

*LOOP ((i,j), P(i,j)=min(P(i,j),3)); {capacitated payoff}

*LOOP ((i,j), if (P(i,j)>0,
                         P(i,j)=1;
                    else
                         P(i,j)=0;)
); {binary payoff}


VARIABLES
         v1   value of game for defenders
         v2   value of game for attackers
         x(i) Attacker marginal distribution on # attackers used
         y(j) Defender marginal distribution on # defenders used
         xcount(i) Attacker marginal dist. * N
```

```
                {= # times i attackers used}
        ycount(j) Defender marginal dist. * N
                {= # times j defenders used}
        a       x-intercept
        b       slope
        c       y-intercept
        d       slope;

POSITIVE VARIABLES x, y, b, d ;

INTEGER VARIABLES xcount, ycount ;
*POSITIVE VARIABLES xcount, ycount ;

EQUATIONS
        objective1        objective funtion
        expreturn_y(i)    expected return
        meandefenders     constraint on mean number of defenders
        probsum_y         sum of probabilities is 1
*       extraconstraint0  extra contraits to help explore
                  different optimal solutions
        ycountdef(j)      definition of ycount
;

objective1.. v1 =e= N*c + d*F ;

expreturn_y(i).. sum(j, P(i,j)*y(j)) =l= c+d*(ord(i)-1);

meandefenders.. N*sum(j,y(j)*(ord(j)-1)) =l= G ;

probsum_y.. sum(j,y(j)) =e= 1 ;

*extraconstraint0.. y('10defend') =e= .25 ;

ycountdef(j)..  ycount(j) =e= N*y(j) ;

MODEL Defense  /objective1,
                expreturn_y,
                meandefenders,
                probsum_y
                ycountdef
/;

EQUATIONS
         objective2       objective funtion
         expreturn_x(j)   expected return
         meanattackers    constraint on mean number of attackers
         probsum_x        sum of probabilities is 1
         xcountdef(i)     definition of xcount
;

objective2 .. v2 =e= N*a - b*G ;

expreturn_x(j) .. sum(i, x(i)*P(i,j)) =g= a-b*(ord(j)-1);

meanattackers .. N*sum(i,x(i)*(ord(i)-1)) =l= F ;
```

```
probsum_x .. sum(i,x(i)) =e= 1 ;

xcountdef(i)..  xcount(i) =e= N*x(i)

MODEL Attack  /objective2,
               expreturn_x,
               meanattackers,
               probsum_x
               xcountdef
/;


SOLVE Defense using mip minimizing v1;

SOLVE Attack using mip maximizing v2;

Time=Time+Defense.resusd+Attack.resusd; {computation time}

DISPLAY v1.l, v2.l, y.l, x.l;
DISPLAY ycount.l,xcount.l, Time;
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Alan R. Washburn, 1994. Two-Person Zero-Sum Games, Institute for Operations Research and the Management Sciences. pp. 36-38

Alan Washburn, 2001, A New kind of fictitious play, Naval Postgraduate School, http://diana.or.nps.mil/~washburn/ModFicPlay/ pp. 1-2

Brown, G.W., 1951, "Iterative Solutions of Games by Fictitious Play." In T.C. Koopmans (ed.), Activity Analysis of Production and Allocation, Cowles Commission Monograph 13, pp. 377-380

Eagle, J. N. and Washburn, A.R. 1991. Cumulative Search-Evasion Games. Naval Research Logistics, Vol. 38, pp.495-510

Robinson, J., 1951. An Iterative Method of Solving a Game. Annals of Mathematics, 54, pp. 296-301

von Neumann, J., and O. Morgenstern. 1944. Theory of Games and Economic Behavior. Princeton U. Press

Winston, W. I., 1991. Operations Research Applications and Algorithms, 2nd ed., PWSKENT, Co.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor James N. Eagle
   Naval Postgraduate School
   Monterey, California

4. Professor Carlyle W. Matthew
   Naval Postgraduate School
   Monterey, California