Capture Probability Over Time in TrapGrid: Comparing Original and an Alternative Algorithm

Nicholas C. Manoukis*

September 9, 2020

1 Background

During discussions starting on 2020-07-28 with colleagues at CSIRO (Matt Hill and Peter Caley) I was asked how TrapGrid[1] calculates average escape probability daily, in detail. Dr. Hill shared results showing that changing the number of insects in the simulation did not affect the mean *capture* probability, but it was showing an effect on the variance (Fig. 1). This was confusing since a lower number of insects should, intuitively, reduce *detection* probability.

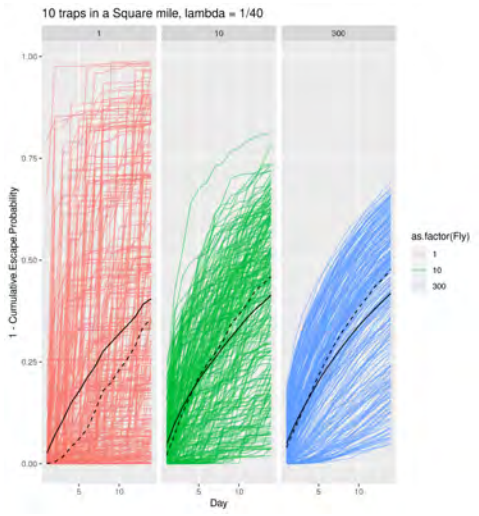

Figure 1: Mean and individual run capture probability for simulations with 1, 10, or 300 insects (Data & Figure: M. Hill)

*USDA-ARS / TCCPRU Hilo, Hawai'i USA — nicholas.manoukis@usda.gov

In the years since TrapGrid was released I have generally thought of the cumulative escape probability generated by TrapGrid as equivalent to *the probability of zero captures*; a strict definition of non-detection. However, review of the manuscript describing the trapping model does not use this definition. Rather, TrapGrid as originally formulated calculated the *Average daily escape probability* and then cumulative values thereof over time[1].

2 Objective

Detail how TrapGrid calculates average daily and cumulative escape probability and compare with an alternative calculation of the probability of zero captures (strict detection).

3 Methods

To compare results I will focus on re-creating the main points of Fig. 6 in Manoukis et al [2]. While looking at these I noticed that they were generated without a random seed value- this feature may not have been implemented at the time. So while I have the original binary (TrapGrid_18feb14.jar) I can't exactly reproduce its output. For this reason, and to eliminate possible additional confusion, I will use the current version of TrapGrid from the github website (`http://bruab.github.io/TrapGrid/`) to represent the "original" algorithm (binary name: TrapGrid.06_24_2015.jar). In this document I will call this "TGO". The modified version ("TrapGrid-Alternative") I will refer to as "TGA". Finally, I will generate results comparable to Fig. 1 with TGA to see how each behaves in detail.

3.1 Original TrapGrid Algorithm

For each "day" of the simulation, the TGO algorithm calculates the *average escape probability* for a set of insect positions. Positions are (usually) calculated via diffusion based on time since start and D, the diffusion coefficient. Once the average for the current day is calculated, the cumulative escape probability to the current day is calculated via the product of the previous day's cumulative value and the current day's average escape probability. In pseudo-code:

[1]The closest I could find on a definition of detection in the original paper is: "Figure 6 shows changes in the *average capture probability* for a single individual medfly as the number of traps per square mile and the attractiveness (driven by use of different lures, for example) of the traps are varied" (Emphasis mine). In retrospect, this could have been made much clearer in the paper.

```
for each simulation;
 for each day;
  escapeProbability = 0;
  positions = distributeInsectPositions;
  for each position;
   escapeProbability = escapeProbability + escapeProbability(position);
  end for;
  dailyEscapeProbability = escapeProbability/numberInsectPositions;
  cumulativeEscapeProbability = cumulativeEscapeProbability * dailyEscapeProbability;
 end for;
 addResultsToOtherSimulations;
end for;
```

An interesting feature I noticed is that the initial escape probability for the day is 0, to accommodate averaging. This might lead to edge cases (where a single insect is used, for example, and it is randomly placed in the same position as a trap so its escape probability is also 0) where the program might throw a div error. This is vanishingly unlikely.

3.2 Alternative Algorithm

For each day of the simulation, the TGA algorithm calculates the *probability all insects escape*, using positions calculated the same way as TGO, via the product of the individual insect escape probabilities. Cumulative probability over days is calculated as above. Pseudo-code:

```
for each simulation;
 for each day;
  escapeProbability = 1;
  positions = distributeInsectPositions;
  for each position;
   escapeProbability = escapeProbability * escapeProbability(position);
  end for;
  cumulativeEscapeProbability = cumulativeEscapeProbability * escapeProbability;
 end for;
 addResultsToOtherSimulations;
end for;
```

Here the initial assumption is all flies escape, and that probability is modified by the addition of insect positions relative to traps. This is intuitively appealing if we are interested in the probability of capturing one or more insects (detection).

4 Results

Systematic comparison employing each of the two algorithms, 6 levels of trap density (3, 5, 9, 25, 49, and 100 traps per square mile), 5 levels of trap attraction (λ between 0.200 and 0.0125) and two insect population sizes ($N = 3$ or 300) with 250 repetitions each results in a total of 30,000 simulations. I did not include 16 traps as in the original Fig 6. This resulted from a scripting error, which meant the simulations did not run- but after compiling the results and generating a figure I realized there was a benefit to excluding at least one set and so did not run them. Averages across the 250 simulations are shown in Fig. 2.

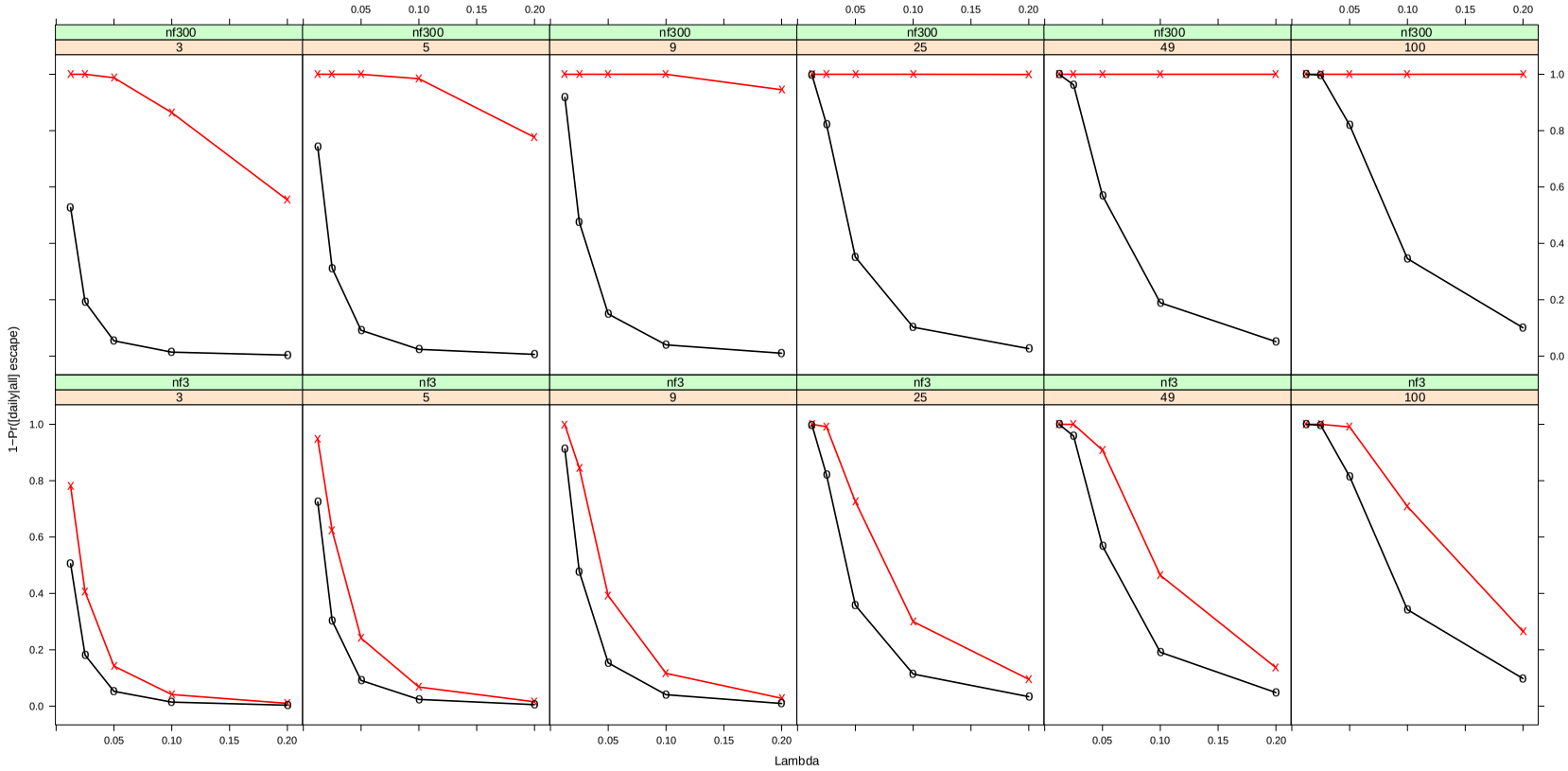

Figure 2: Average capture probability as a function of trap attraction (λ, abscissa), number of traps (top row panel labels), number of insects (bottom row panel labels, nf300 = 300 insects; nf3 = 3 insects), and algorithm (black "o" = TGO; red "x" = TGA). Capture probability is "daily average" for TGO and "of one or more" for TGA.

Differences between TGO and TGA are clear when the number of insects is 300; at 3 insects, results are largely comparable: there is a similar impact of λ and number of traps under both calculations, and they are not quantitatively very far apart. It is also clear that TGO is insensitive to the number of insects in terms of average expectation- this is in agreement with Fig. 1.

I plotted individual run results and means in Fig. 3 to see how the two algorithms behave relative to numbers of insects. TGO behaves as we have seen: fewer insects increases variation between runs, but the means are unaffected. For TGA there is overall more variation compared with TGO, and it is especially pronounced when there are 3 insects compared with 300; means are strongly affected by the number of insects.

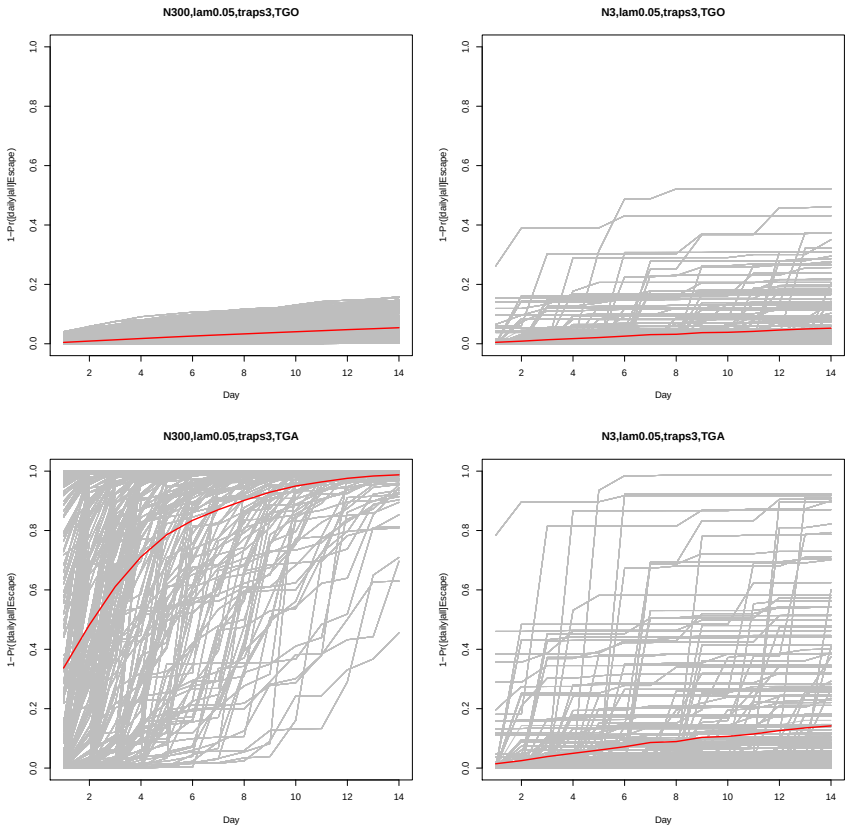

Figure 3: Relationship between day and capture, individual runs (grey lines) and mean (red lines) for two algorithms (TGO and TGA) and two numbers of insects ($N = 3$ or 300). Shared parameters: $\lambda = 0.05$, Number of traps $= 3$

5 Conclusion

In no particular order, here are the main conclusions I drew from this exercise:

1. The original TrapGrid (TGO) calculates the *cumulative daily average escape probability* over time. This is not in accordance with the strict definition of detection probability, which I see as the *cumulative daily probability of capturing one or more insects*, but it is related.

2. "Insects" in TGO should be considered "sampling points" rather than literal individuals. The mean result is largely insensitive to the number of sampling points, but the variance is affected.

3. The cumulative average daily capture probability (TGO) is an estimate of trap network sensitivity, and a conservative one. This is not true for TGA, which presents perhaps the "best case scenario" but does not include a lot of important biology that might impact capture/detection (e.g. lure insensitivity, trap efficiency, etc).

4. TGA might be a good approach if you want to strictly model detection of very small populations: Certainly < 10, probably < 5. It should be remembered, however, that it does not give a conservative estimate of capture probability (see above).

5. If you are not certain of the number of insects TGO is a more robust approach to analyzing trap network sensitivity.

I am considering releasing a version of TrapGrid with TGA implemented as well as TGO, selectable at the command line.

References

[1] Brian Hall. TrapGrid by BrianReallyMany. `http://bruab.github.io/TrapGrid/`. Accessed: 2020-09-08.

[2] Nicholas C. Manoukis, Brian Hall, and Scott M. Geib. A computer model of insect traps in a landscape. *Scientific Reports*, 4:7015, 2014.

6 Appendix

Below is the section of Java code I modified to switch from TGO to TGA.

```
/**
 * Main method for simulation; also writes results to output files.
 */
public SimulationResultsHolder runSimulation() {
    resultsHolder = new SimulationResultsHolder();
    resultsHolder.addFlyReleaseInfo(fr.toString());
    for (int i = 1; i<=numberOfDays; i++) {
        System.err.println("Running simulation for day " + i + "...");
        //double totalProbForDay = 0; //TGO
        double totalProbForDay = 1;  //TGA
        //int numberOfFlies = 0;  //TGO
        Iterator<OutbreakLocation> releasePointItr = fr.allOutbreakLocations.iterator();
        while (releasePointItr.hasNext()) {
            OutbreakLocation currentReleasePoint = releasePointItr.next();
            ArrayList<Point2D.Double> flyLocations = currentReleasePoint.locateFlies(i);
            Iterator<Point2D.Double> flyLocationItr = flyLocations.iterator();
            while (flyLocationItr.hasNext()) {
                Point2D.Double currentLocation = flyLocationItr.next();
                Double currentEscapeProb = tg.getTotalEscapeProbability(currentLocation);
                String[] results = {Integer.toString(i), currentReleasePoint.shortString(),
                        locationToString(currentLocation), Double.toString(currentEscapeProb
                resultsHolder.addRawData(results);
                //totalProbForDay += currentEscapeProb; //TGO
                totalProbForDay *= currentEscapeProb; //TGA
                //numberOfFlies += 1;  //TGO
            }
        }
        //double avgForDay = totalProbForDay / numberOfFlies; //TGO
        double avgForDay = totalProbForDay; //TGA
        cumulativeProb *= avgForDay;
        resultsHolder.addAvgEscapeProbability(i, avgForDay);
    }
    System.err.println("Simulation complete!");
    return resultsHolder;
}
```