Beyond the Threshold: Investigating the Impact of Recruitment and Quorum Rules in the Robinson Model

Efe Mirkan Guner
Intelligence in Animals and Machines
University of Sussex

January 2026

Contents

Abstract

Collective decision-making in ant colonies can appear sophisticated despite relying on relatively simple individual rules. Previous work has shown that a fixed acceptance threshold is often sufficient to explain accurate nest-site choice without requiring direct comparisons between options. In this report, the Robinson et al. (2011) threshold model is extended by introducing recruitment and quorum mechanisms, which are systematically explored to examine how they interact with perceptual noise in nest quality.

A stochastic agent-based simulation is used to compare three conditions: a baseline threshold model, a model with recruitment only, and a model combining recruitment with a quorum rule. Model performance is evaluated in terms of decision accuracy, decision time, and commitment dynamics under increasing levels of nest-quality noise and across different quorum sizes. The results show that recruitment alone improves robustness to noise, maintaining relatively high accuracy even when individual quality assessments are unreliable. In contrast, adding a quorum rule leads to faster decisions but reduces accuracy under high noise due to premature collective commitment. These findings highlight a clear speed–accuracy trade-off and illustrate how small changes to individual-level rules can act as leverage points that shape collective outcomes.

1 Introduction

Collective decision-making in social animals has attracted significant interest because groups are often able to solve complex problems despite individuals having limited information and simple behavioural rules. Ant colonies are a well-studied example of this phenomenon, particularly in the context of nest-site selection. When faced with multiple potential nest sites, colonies of *Temnothorax* ants are able to choose high-quality options reliably and efficiently, even when individual ants do not appear to make explicit comparisons between alternatives.

Experimental and modelling work suggests that accurate collective decisions can emerge from relatively simple individual-level mechanisms. E. J. H. Robinson, Franks, et al. (2011) proposed a threshold-based model in which individual ants accept a nest if its perceived quality exceeds an internal acceptance threshold. In this model, ants do not need to compare sites directly. Instead, colony-level discrimination emerges from the aggregation of individual accept or reject decisions. Despite its simplicity, this model was shown to reproduce several key features of real ant emigrations, including high accuracy and realistic decision times.

Further empirical evidence supports the idea that collective comparison can arise without individual comparison. E. J. H. Robinson, Feinerman, and Franks (2014) demonstrated that ant colonies can still choose the better nest even when individual ants are experimentally prevented from visiting both options. Ants that experience only poor nests tend to continue searching, while ants that encounter good nests are more likely to commit. These asymmetric individual responses are sufficient to produce reliable colony-level choices. Together, these findings challenge the assumption that sophisticated group decisions require sophisticated individual cognition.

Although threshold models capture important aspects of ant decision-making, real colonies also use additional mechanisms such as recruitment and quorum sensing. Recruitment introduces positive feedback by biasing movement towards sites with more committed individuals, while quorum rules allow colonies to trigger rapid commitment once enough ants support a site. These mechanisms are known to influence both the speed and accuracy of collective decisions, but their interaction with perceptual noise remains less well understood. In particular, it is unclear whether recruitment and quorum rules improve robustness when individual assessments of nest quality are noisy, or whether they can instead amplify early errors.

Computational modelling provides a useful tool for addressing these questions by allowing systematic manipulation of mechanisms and parameters that are difficult to control experimentally. As argued by Guest and Martin (2021), formal models force explicit assumptions and make it possible to explore the consequences of simple rules at the system level. Similar approaches have also been applied to collective decision-making in artificial systems, such as robot swarms, where different information sharing strategies lead to distinct speed-accuracy trade-offs.

In this report, the threshold model of E. J. H. Robinson, Franks, et al. (2011) is extended by incorporating recruitment and quorum mechanisms. Using a stochastic agent-based simulation, three conditions are compared: a baseline threshold model, a model with recruitment only, and a model combining recruitment with a quorum rule. The central aim is to examine how these mechanisms interact with increasing levels of perceptual noise in nest quality. Decision accuracy, decision time, and commitment dynamics are analysed across noise levels and quorum sizes. By treating recruitment strength and quorum size as intervention points, this work also connects to broader systems thinking ideas, where small changes in rules can have large effects on collective outcomes.

2 Methods

2.1 Model Overview

A stochastic Monte Carlo simulation was implemented to reproduce and extend the threshold-based decision model developed by E. J. H. Robinson, Franks, et al. (2011). This modelling approach was selected because nest-site choice in *Temnothorax albipennis* involves uncertainty at both the sensory and behavioural levels. Individual scouts vary in how they perceive nest quality and differ in the thresholds applied when deciding whether to commit to a site. A deterministic approach would be unable to capture the variability in decision times and final nest choices observed in empirical studies. Monte Carlo simulation allows repeated sampling of perceptual noise, heterogeneous thresholds, and probabilistic movement through the environment, enabling an examination of how these sources of uncertainty influence collective outcomes.

In addition to the baseline threshold model, two extensions were examined. The first introduced recruitment, allowing ants to bias movement towards nests with higher numbers of committed individuals. The second combined recruitment with a quorum rule, where a colony decision was defined by the time at which a specified number of ants had committed to the same nest.

2.2 Environment and State Space

The simulated environment consisted of three discrete nest states: the original home nest, which was assigned a quality of negative infinity to ensure it was never selected; a poor nest of fixed quality 4; and a good nest of fixed quality 6.

Site transitions were governed by a fixed 3×3 probability matrix P, where column j represents the current site and row i represents the next site. The structure of the environment and the baseline transition probabilities between nests are illustrated in Figure 1.

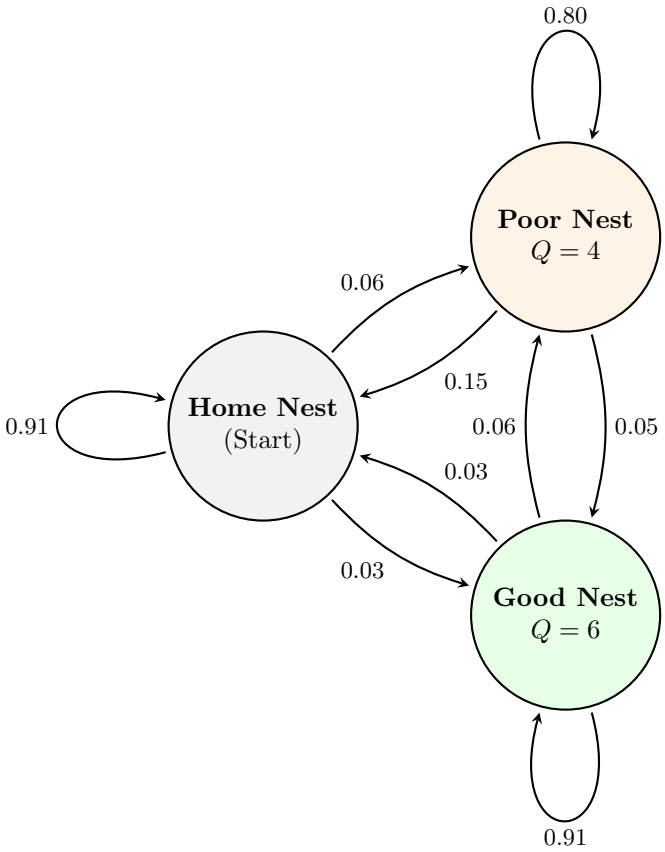

Figure 1: **System diagram of the nest-selection environment.** Agents begin at the Home nest and move stochastically between sites according to the baseline transition probabilities shown. The Poor and Good nests have fixed qualities $Q = 4$ and $Q = 6$ respectively. Probabilities correspond to the baseline transition matrix P before any recruitment bias is applied.

The transition matrix P used in the simulation is given explicitly by:

$$P = \begin{pmatrix} 0.91 & 0.15 & 0.03 \\ 0.06 & 0.80 & 0.06 \\ 0.03 & 0.05 & 0.91 \end{pmatrix}$$

Travel times between nests were drawn from normal distributions. The means were defined by a corresponding matrix T_{mean}, and standard deviations were set to one-fifth of the respective mean:

$$T_{\text{mean}} = \begin{pmatrix} 1 & 36 & 143 \\ 36 & 1 & 116 \\ 143 & 116 & 1 \end{pmatrix}$$

2.3 Individual Decision Process (Threshold Model)

The simulation modelled agents sequentially. Each agent began at the home nest. At every visited site, the perceived quality was generated by sampling from a normal distribution centred on the nest's true quality:

$$\hat{Q}_i = Q_i + \mathcal{N}(0, \sigma_q)$$

where Q_i is the true quality and σ_q controls the level of perceptual noise.

At the start of each simulation, every agent was assigned an internal acceptance threshold drawn from a normal distribution:

$$\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta)$$

where $\mu_\theta = 4.5$ and $\sigma_\theta = 0.6$. When an agent visited a site, it compared the sampled perceived quality against its threshold. If $\hat{Q}_i \geq \theta$, the site was accepted, and the decision time was recorded. If the condition was not met, the agent continued searching. The simulation for each ant continued until a nest was accepted or until 1000 movement steps were completed.

Algorithm 1: Simulation of a single ant in the threshold model

```
Initialize the agent at the home nest (Site 0);
Sample an acceptance threshold θ from the population distribution N(μ_θ, σ_θ);
while no nest has been accepted and step count < 1000 do
    Sample perceived quality Q̂ for the current nest;
    if Q̂ ≥ θ then
        Accept the current nest;
        Record the current time as the decision time;
        break;
    else
        Select the next nest using the transition probability matrix P;
        Sample a travel time from T_mean and update the accumulated decision time;
        Update current location;
```

2.4 Recruitment Mechanism

When recruitment was enabled, the movement of scouting agents was biased towards nests with larger numbers of previously committed ants. In this sequential simulation, the decisions of earlier ants influenced the movement probabilities of later ants. Let p_i be the baseline probability of moving to nest i, and R_i be the number of ants already committed to that nest. The biased probability p_i' was calculated as:

$$p_i' \propto p_i + rR_i$$

where r is the recruitment strength parameter ($r = 0.05$ when enabled). This introduced positive feedback at the colony level by increasing the likelihood that subsequent scouts would encounter popular nests. Importantly, recruitment only modified movement probabilities; the individual-level acceptance rule remained unchanged.

2.5 Quorum-Based Interpretation of Colony Decisions

In the quorum condition, the decision rule was applied as a post-hoc analysis of colony dynamics rather than as an online termination condition. During the simulation, the time at which each ant committed to a nest was recorded. After all ants had been simulated, the acceptance times for each nest were analysed to determine whether a quorum had been reached.

For a given quorum size k, the acceptance times associated with each nest were sorted chronologically. If a nest accumulated at least k committed ants, the colony decision time was defined as the acceptance time of the k-th ant to arrive at that nest. This post-hoc method accurately reflects how a colony "locks in" to a decision without requiring individual ants to have global knowledge of the quorum count.

Algorithm 2: Sequential simulation with recruitment and quorum analysis

```
Initialize recruiter count for each nest j (R_j ← 0);
Initialize empty lists T_j to store acceptance times for each nest;
for ant i ← 1 to n do
    Run Algorithm 1 (Agent Decision Process);
    if Ant commits to nest j then
        Append current time to acceptance list T_j;
        Increment recruiter count (R_j ← R_j + 1);
    if Recruitment enabled then
        Update transition probabilities P based on current R;
foreach nest j do
    Sort acceptance times T_j in ascending order;
    if |T_j| ≥ k (where k is quorum size) then
        Colony decision time ← the k-th value in T_j;
```

2.6 Parameter Sweep and Experimental Design

The study examined how two distinct sources of uncertainty and constraint influenced collective decision accuracy and dynamics:

1. **Nest-quality noise (σ_q)**
   We swept nest-quality noise σ_q across a broad range $(0.1, 0.5, 1.0, 1.5, 3.0, 4.0)$. Following the noisy quality perception used in the Robinson-style modelling framework E. J. H. Robinson, Franks, et al. (2011), increasing σ_q increases overlap between the perceived quality distributions, making correct discrimination less reliable. This sweep tests the decision mechanism across regimes ranging from clear discriminability to high uncertainty.

2. **Quorum size (k)**
   In conditions where the quorum mechanism was enabled, quorum thresholds k were swept from small ($k = 5$) to large ($k = 80$) values. These values span a range of effective quorum sizes to systematically test the speed–accuracy trade-off often observed in collective decision-making E. J. H. Robinson, Feinerman, and Franks (2014). Smaller quorums mimic urgent decision-making, triggering rapid lock-in at the risk of error, whereas larger quorums enforce broader consensus.

Recruitment Strength (r): Recruitment strength was set to $r = 0.05$ to introduce a conservative positive feedback loop. This small bias ensures that private assessments continue to dominate early decisions, allowing recruitment to amplify agreement only once a preference begins to emerge. In this way, recruitment acts as weak positive feedback rather than forcing near-deterministic convergence.
Simulation Protocol: For the noise analysis, three conditions (Baseline, Recruitment Only, Recruitment + Quorum at fixed $k = 20$) were tested across all σ_q levels. To investigate the speed–accuracy trade-off specifically, a separate parameter sweep varied quorum size k from 5 to 80 at a fixed moderate noise level ($\sigma_q = 1.5$). For each parameter combination, 30 independent colony replicates were simulated.

Table 1: Summary of model parameters

Parameter	Symbol	Value	Role in model
Number of ants	n	500	Controls colony size
Nest qualities	Q_g, Q_p	6, 4	Defines decision alternatives
Quality noise	σ_q	0.1 – 4.0	Controls perceptual uncertainty (Variable)
Mean threshold	μ_θ	4.5	Sets overall choosiness of ants
Threshold SD	σ_θ	0.6	Introduces individual variability
Recruitment strength	r	0.05 (when enabled)	Controls bias towards popular nests
Quorum size	k	5 – 80	Determines when colony commits (Variable)
Travel time	T_{ij}	Matrix	Mean travel time between sites

2.7 Output Measures and Visualisation

Three summary measures were produced for each experimental condition:

1. **Decision Accuracy:**
   Defined as the proportion of colony replicates in which the "Good nest" (quality 6) was selected as the final choice.

2. **Decision Time:**
   For baseline and recruitment-only conditions, this was measured as the mean time taken for all individual ants to commit. For the quorum condition, it was defined as the time step at which the k-th ant committed to the winning nest.

3. **Commitment Dynamics:**
   The cumulative number of ants committed to each nest was tracked over time for representative runs to visualise the emergence of consensus.

3 Results

3.1 Accuracy as a Function of Nest Quality Noise

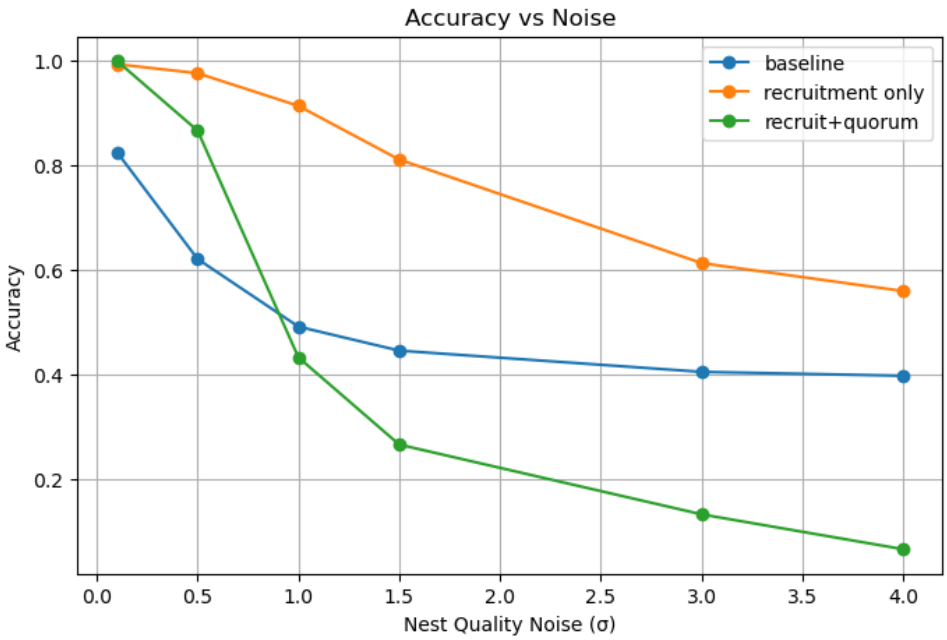

Figure 2: **Decision accuracy vs. nest quality noise (σ)**. Comparison of the baseline model (blue), recruitment only (orange), and recruitment with quorum (green). Accuracy is defined as the proportion of replicates in which the colony selected the Good nest ($Q = 6$).

Figure 2 shows accuracy as a function of nest quality noise (σ) for the baseline, recruitment-only, and recruitment + quorum conditions.

In the baseline condition (blue line), which implements a simple individual acceptance rule without recruitment or quorum enforcement, decision accuracy decreases steadily as perceptual noise increases. At low noise levels, colonies reliably select the high-quality nest, but accuracy declines toward chance levels as noise increases, indicating reduced reliability under uncertain quality assessment. This behaviour is consistent with previous implementations of threshold-based decision-making under noisy conditions (E. J. H. Robinson, Franks, et al. 2011).

Introducing recruitment (orange line) is associated with higher accuracy across the full range of noise values. Accuracy remains close to 1.0 at low noise and declines more gradually than in the baseline condition, remaining well above chance even at the highest noise levels tested.

When recruitment is combined with a quorum rule (green line), accuracy is high under low-noise conditions but deteriorates rapidly as noise increases. Beyond moderate noise levels ($\sigma > 1.0$), accuracy drops sharply and becomes lower than both the baseline and recruitment-only conditions. At high noise levels, colony decisions frequently converge on the inferior nest, indicating increased sensitivity to early stochastic commitments.

3.2 Effect of Quorum Size on Accuracy

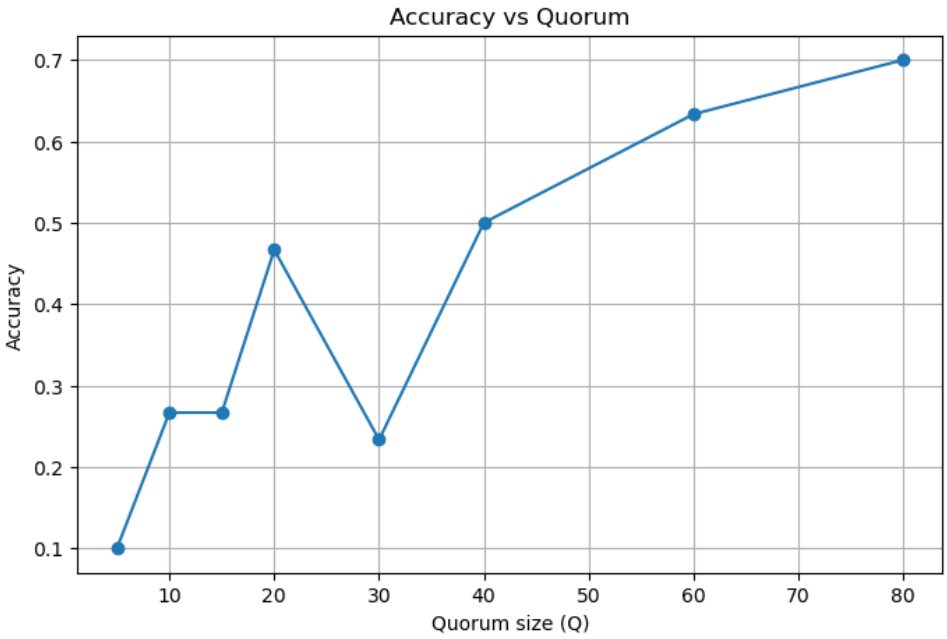

Figure 3: **Accuracy vs. quorum size (k).** The relationship between the quorum threshold and decision accuracy at a fixed noise level of $\sigma_q = 1.5$. Data points represent the mean of 30 independent replicates.

Figure 3 shows accuracy as a function of quorum size (k) at a fixed level of nest quality noise ($\sigma_q = 1.5$).

Accuracy varies systematically with quorum size. Very small quorum thresholds ($k = 5$) are associated with extremely low accuracy (approximately 10%). As quorum size increases, accuracy generally improves, with larger quorum thresholds producing more reliable nest selection.

The relationship between quorum size and accuracy is not strictly monotonic at small to intermediate values of k. A noticeable fluctuation occurs between $k = 20$ and $k = 30$, where accuracy temporarily decreases relative to neighbouring quorum sizes. This variability indicates increased outcome variability at intermediate quorum thresholds under moderate noise conditions.

At the largest quorum sizes tested ($k = 80$), accuracy reaches its highest level (approximately 70%). Overall, larger quorum thresholds are associated with higher decision accuracy at this noise level.

3.3 Decision Speed and Trade-offs

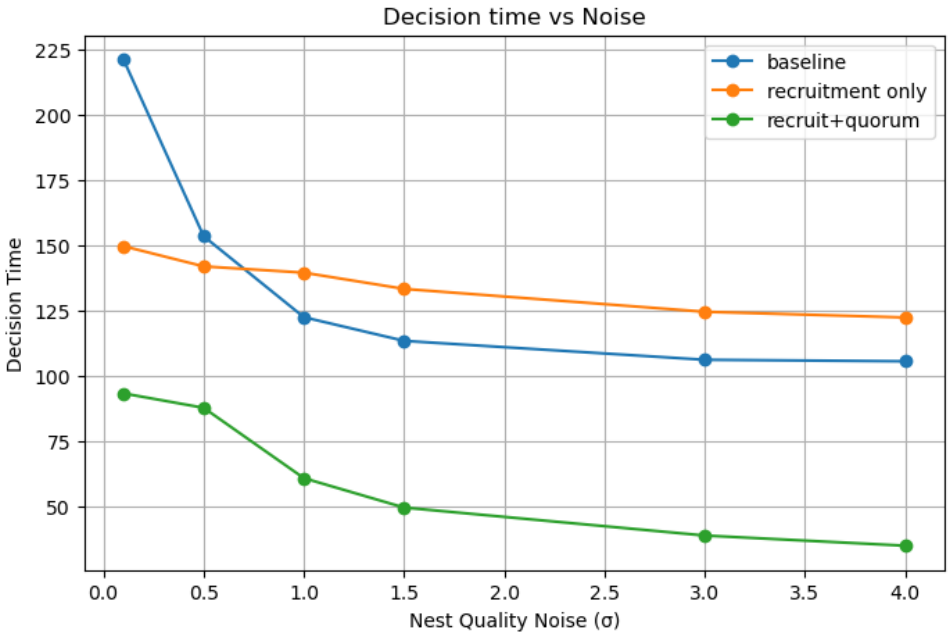

Figure 4: **Mean decision time as a function of noise**. Comparison of baseline (blue), recruitment-only (orange), and recruitment + quorum (green) conditions. For the quorum condition, decision time is defined as the time taken to reach k committed ants.

Figure 4 shows the mean decision time as a function of nest quality noise for the three model conditions.

Across all conditions, decision time decreases as perceptual noise increases. In the baseline condition, decision times are longest at low noise levels and decline steadily as noise increases.

Recruitment generally reduces decision time relative to the baseline, particularly at low to moderate noise levels. The recruitment + quorum condition produces the shortest decision times overall, with the time to consensus decreasing sharply as noise increases. At high noise levels, the quorum mechanism drives the colony to a decision in fewer than 50 time steps, compared to over 100 steps for the baseline.

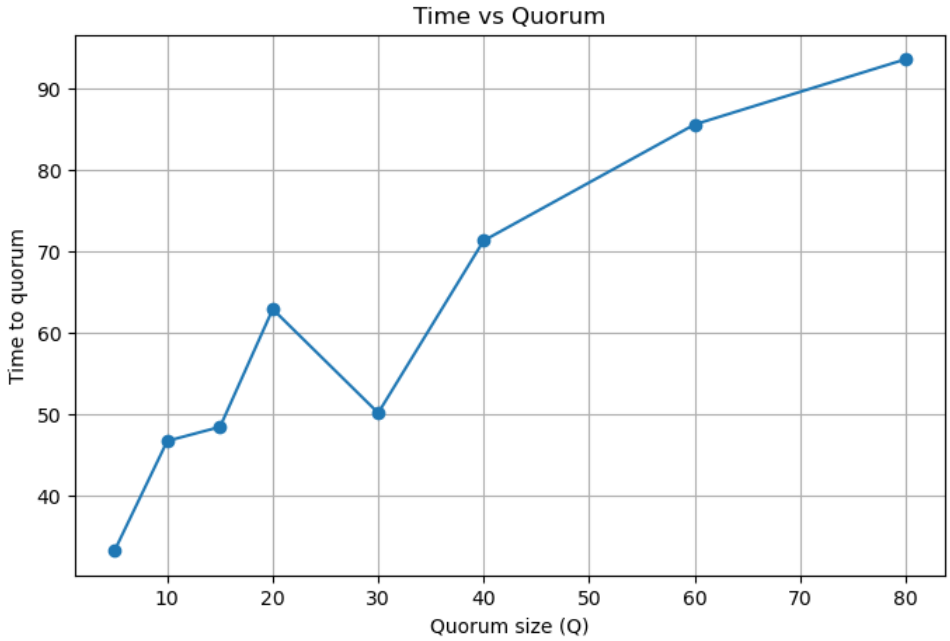

Figure 5: **Time to quorum vs. quorum size (k).** Relationship between the quorum threshold and the time taken to reach a collective decision at $\sigma_q = 1.5$.

However, this increase in speed is associated with a tunable cost. Figure 5 shows that the time required to reach a decision increases approximately linearly with quorum size (k). The time to consensus rises from approximately 35 steps at $k = 5$ to over 90 steps at $k = 80$. This demonstrates that the quorum threshold directly controls decision speed, with larger quorums producing slower collective commitment.

3.4 Commitment Dynamics

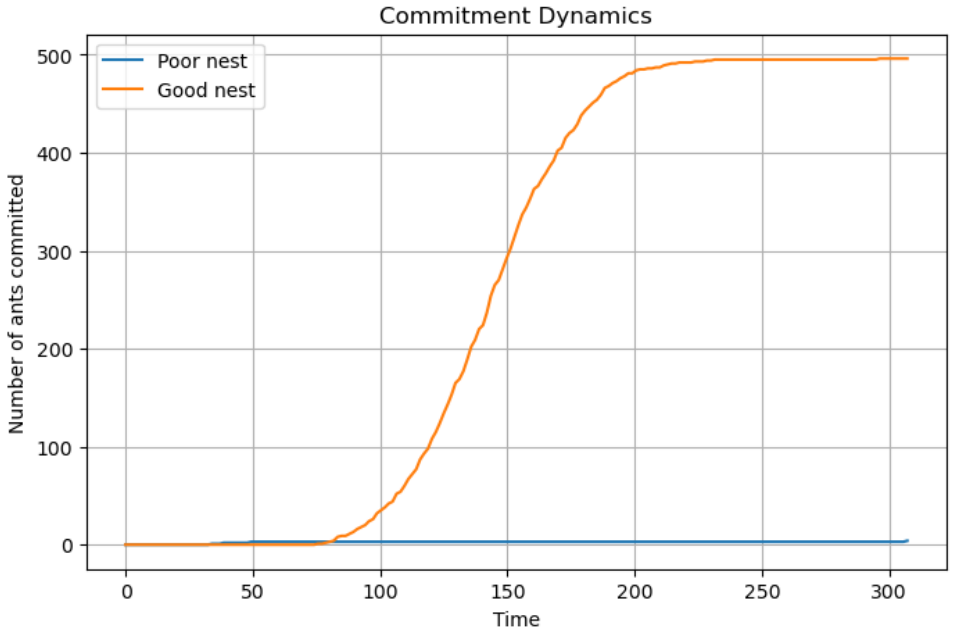

Figure 6: **Temporal dynamics of commitment**. The cumulative number of ants committed to the Good nest (orange) and Poor nest (blue) over time in a single representative simulation. The system exhibits a non-linear transition to consensus.

Figure 6 illustrates the temporal dynamics of ant commitments to each nest for a representative simulation run.

Commitment to the high-quality nest initially increases slowly, followed by a rapid acceleration once a sufficient number of ants have committed. This produces a sigmoidal growth pattern, with a sharp transition from gradual accumulation to rapid convergence.

In contrast, commitment to the poor-quality nest remains low throughout the simulation and does not exhibit sustained growth. The divergence between the two nests becomes increasingly pronounced over time, with the high-quality nest rapidly dominating collective commitment.

These dynamics reflect the emergence of colony-level comparison through individual acceptance and recruitment behaviour, without requiring direct comparison of nest options by individual ants (E. J. H. Robinson, Franks, et al. 2011).

4 Discussion

4.1 Robustness and Noise in Threshold-Based Collective Decisions

The results show that perceptual noise is a binding constraint on decision reliability, but its effect depends on the interaction mechanism. In the baseline condition, accuracy declined monotonically with increasing sensory uncertainty, indicating that without communication the colony is limited by individual sampling precision. Recruitment improved robustness, sustaining high accuracy through moderate noise. In contrast, adding a quorum rule produced a brittle regime that collapsed under high-noise conditions (Figure 2).

This pattern is consistent with the threshold mechanism in E. J. H. Robinson, Franks, et al. (2011), where acceptance is driven by whether a noisy quality sample exceeds an internal threshold. At low noise, the probability mass remains sufficiently separated for reliable discrimination. As σ_q increases, erroneous accepts of the poor nest and erroneous rejects of the good nest become common, pushing behaviour toward chance. Because agents act independently in the baseline model, these errors do not self-correct and accumulate into the observed decline.

The recruitment-only condition suggests that positive feedback can function as a collective noise filter. By biasing transitions toward sites with existing commitments, recruitment increases coherence once an initially correct majority forms. In this sequential simulation, early commits influence many subsequent moves; when early commits reflect true quality (typical at low–moderate σ_q), recruitment reinforces the correct signal and dampens later mis-samples.

The failure mode of recruitment-plus-quorum at high noise highlights the cost of amplifying early stochasticity. For $\sigma_q 1.5$, early commitments are often driven by random fluctuations, and with r held fixed the same feedback strength is applied even when signal quality is degraded. The quorum is implemented as a post-hoc trigger on the instantaneous commitment count, so once the threshold is reached the colony locks in immediately. Under these conditions, small early imbalances are converted into premature lock-in to the suboptimal nest; near the transition region, brief run-to-run irregularities (plateaus or reversals) were also observed.

4.2 Speed–Accuracy Trade-offs and Quorum Thresholds

A central hypothesis of this study was that quorum thresholds mediate a speed–accuracy trade-off in collective decision-making. The results support this: increasing perceptual noise produced faster decisions across all variants, and quorum rules shifted the balance between speed and reliability. In particular, quorum-based decisions were consistently faster than baseline decisions under high-noise conditions, but this speed came with reduced accuracy.

Varying quorum size clarifies the mechanism. Larger thresholds produced slower decisions, whereas small quorums enabled rapid consensus. In this implementation, quorum size acts as a control on evidence accumulation: small k commits after limited support, while large k delays commitment until broader convergence. Because updates are sequential, early leads can cascade into further commitments, so smaller quorums are reached disproportionately quickly.

Crucially, faster decisions here do not imply better information processing. Under high σ_q, early threshold crossings are often driven by stochastic fluctuations rather than reliable quality differences. With r held fixed, the same feedback intensity applies even as signal quality deteriorates, increasing the chance that noise-generated leads are reinforced. The quorum rule is applied as a post-hoc trigger on the instantaneous commitment count, so the process terminates immediately at threshold without an explicit correction phase. This combination explains why quorum decisions can be fast yet inaccurate when uncertainty is high.

These patterns match empirical observations that colonies vary quorum thresholds with urgency and risk (E. J. Robinson et al. 2009; E. J. H. Robinson, Feinerman, and Franks 2014). In stable contexts, larger quorums promote accuracy; in time-pressured conditions, smaller quorums enable rapid relocation at the expense of optimal choice. Here, the same qualitative trade-off arises from tuning k, though the exact turning point depends on sequential update order and the post-hoc quorum trigger.

Overall, the speed–accuracy trade-off observed here highlights quorum size as a collective control knob regulating when the colony stops sampling and commits under uncertainty.

4.3 Recruitment, Positive Feedback, and Commitment Dynamics

The commitment dynamics in Section 3.4 provide mechanistic insight into how recruitment and quorum rules shape collective outcomes. The sigmoidal accumulation of commitments to the high-quality nest reflects positive feedback: early commitments increase the probability of further commitments to the same option. This non-linear growth suggests that collective decisions arise not only from continued sampling, but from feedback-driven amplification once an early lead forms.

Recruitment drives this shift by biasing movement toward nests that already contain committed agents. The colony therefore transitions from an exploratory phase, with sparse and reversible commitments, to a convergence phase in which commitment becomes self-reinforcing. In the sequential update scheme used here, this transition can occur abruptly because the influence of early commits propagates immediately to later agents, producing a steep rise in the commitment curve. The poor-quality nest rarely sustains comparable growth, despite individuals never directly comparing options, consistent with prior work (E. J. H. Robinson, Franks, et al. 2011; E. J. H. Robinson, Feinerman, and Franks 2014).

These dynamics also clarify why recruitment improves robustness under moderate noise but contributes to failure at high noise. When early commitments are informative, positive feedback amplifies the correct option and suppresses later mis-samples. When early commitments are largely stochastic, the same mechanism amplifies fluctuations rather than signal. Occasional run-to-run reversals in the early phase indicate that small initial differences can be decisive, especially with r fixed across noise regimes.

Quorum rules sharpen this dynamic by imposing a hard stopping condition. In this implementation, the quorum is applied as a post-hoc trigger on the instantaneous commitment count, so once the threshold is reached the process terminates immediately, leaving little scope for correction. Combined with recruitment, this converts early commitment patterns into irreversible outcomes, explaining premature lock-in under high noise.

Overall, recruitment and quorum mechanisms shape the temporal structure of decision-making, determining whether feedback stabilises accurate choices or amplifies noise (Guest and Martin 2021).

4.4 Leverage Points in the Robinson Model

The results can be interpreted through systems thinking, particularly leverage points as articulated by Meadows (1999) and expanded in Meadows and Wright (2011). Leverage points are places where small structural or parametric changes yield disproportionate shifts in system behaviour. Viewed this way, the Robinson model exposes distinct intervention points that reshape collective outcomes.

A key leverage point is recruitment strength. Recruitment changes information flow by introducing positive feedback between individual commitments. In this implementation, with r held fixed across noise regimes, the same amplification operates even when signal quality deteriorates. As shown in Sections 4.1 and 4.3, recruitment improves robustness under moderate noise by reinforcing early correct commitments, yet under high noise it can amplify chance early leads, producing rapid convergence on an arbitrary option. Occasional run-to-run irregularities near the transition are consistent with finite-sample stochasticity interacting with sequential updates, but the qualitative shift in behaviour remains clear.

A second leverage point is the quorum threshold. Quorum size controls when the decision process terminates, setting the balance between exploration and commitment. As shown in Section 4.2, small quorums favour rapid decisions but increase susceptibility to noise, whereas larger quorums delay commitment and improve accuracy. Here the quorum is applied as a post-hoc trigger on the instantaneous commitment count, so crossing the threshold ends the process immediately rather than allowing an extended correction window; this design choice strengthens the stopping-rule effect and helps explain the sharp lock-in observed at high σ_q.

Importantly, neither recruitment nor quorum requires agents to represent uncertainty explicitly or perform complex comparisons. System-level properties such as robustness, speed, and fragility emerge from the interaction of simple behavioural rules with feedback and stopping conditions, supporting the view that computational models help build theory by exposing how qualitative changes arise from minimal assumptions (Guest and Martin 2021).

Alternative approaches, such as Bayesian swarm models, encode uncertainty and belief updating at the agent level (Ebert et al. 2020). While such models can achieve strong inferential accuracy, they rely on heavier cognitive and informational assumptions. In contrast, the present threshold-based framework reproduces key phenomena (speed–accuracy trade-offs and noise sensitivity) without explicit probabilistic reasoning, highlighting the explanatory value of simple mechanisms.

Overall, interpreting the Robinson model via leverage points clarifies why small changes to recruitment strength or quorum thresholds can qualitatively reshape behaviour. These parameters do not merely fine-tune outcomes; they alter how the system responds to uncertainty and early stochasticity.

4.5 Limitations and Future Directions

Despite demonstrating how recruitment and quorum mechanisms shape robustness, speed, and commitment dynamics, the present model has several limitations that suggest directions for future work. Most notably, the model assumes that agents have no memory of past nest evaluations and that acceptance thresholds remain fixed throughout the decision process. As a result, ants cannot adapt their behaviour based on prior experience or adjust their selectivity in response to changing environmental conditions. While this simplification is consistent with the minimal assumptions of the Robinson framework, it limits the model's ability to capture longer-term learning or context-dependent decision strategies.

Another limitation concerns the treatment of noise as a static parameter. In natural settings, perceptual uncertainty may vary across individuals, over time, or in response to environmental cues. Previous exploratory modelling of the threshold rule has shown that variability in noise and threshold distributions alone can substantially influence collective outcomes, even in the absence of recruitment or quorum mechanisms (Guner 2025). Extending the present model to incorporate dynamic or heterogeneous noise profiles would allow a more nuanced examination of how interaction rules respond to fluctuating uncertainty.

Future work could also explore adaptive quorum thresholds or state-dependent recruitment strength. In biological systems, quorum size is not necessarily fixed but may depend on urgency, predation risk, or nest availability. Allowing quorum thresholds to vary dynamically could enable the colony to balance speed and accuracy more flexibly, potentially mitigating the premature lock-in observed under high-noise conditions. Similarly, introducing limits or decay in recruitment strength could prevent excessive amplification of early stochastic commitments.

Finally, the current model focuses on a simple two-option nest choice scenario. Extending the framework to multiple competing nests or spatially structured environments would provide a more realistic test of its scalability and robustness. Such extensions would help clarify whether the leverage points identified here remain effective as task complexity increases.

Overall, while the model deliberately prioritises simplicity, its limitations point toward clear and tractable extensions. Addressing these would deepen our understanding of how minimal behavioural rules give rise to adaptive collective decision-making under uncertainty.

References

Ebert, Julia T. et al. (2020). "Bayes Bots: Collective Bayesian Decision-Making in Decentralized Robot Swarms". In: *PLOS Computational Biology* 16.8, e1007959. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007959.

Guest, Olivia and Andrea E. Martin (2021). "How computational modeling can force theory building in psychological science". In: *Perspectives on Psychological Science* 16.4, pp. 789–802. DOI: 10.1177/1745691620970585.

Guner, Efe Mirkan (2025). *Exploring the Limits of the Threshold Rule: Noise, Variability, and Collective Nest Choice*. University of Sussex.

Meadows, Donella H. (1999). *Leverage Points: Places to Intervene in a System*. The Sustainability Institute. URL: https://donellameadows.org/archives/leverage-points-places-to-intervene-in-a-system/.

Meadows, Donella H. and Diana Wright (2011). *Thinking in systems: a primer*. Nachdr. White River Junction, Vt: Chelsea Green Pub. 218 pp. ISBN: 978-1-60358-055-7.

Robinson, Elva J. H., Ofer Feinerman, and Nigel R. Franks (July 22, 2014). "How collective comparisons emerge without individual comparisons of the options". In: *Proceedings of the Royal Society B: Biological Sciences* 281.1787, p. 20140737. ISSN: 0962-8452, 1471-2954. DOI: 10.1098/rspb.2014.0737. URL: https://royalsocietypublishing.org/doi/10.1098/rspb.2014.0737 (visited on 01/14/2026).

Robinson, Elva J. H., Nigel R. Franks, et al. (May 24, 2011). "A Simple Threshold Rule Is Sufficient to Explain Sophisticated Collective Decision-Making". In: *PLoS ONE* 6.5. Ed. by Frederick R. Adler, e19981. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0019981. URL: https://dx.plos.org/10.1371/journal.pone.0019981 (visited on 11/26/2025).

Robinson, Elva J.H. et al. (July 22, 2009). "Do ants make direct comparisons?" In: *Proceedings of the Royal Society B: Biological Sciences* 276.1667, pp. 2635–2641. ISSN: 0962-8452, 1471-2954. DOI: 10.1098/rspb.2009.0350. URL: https://royalsocietypublishing.org/doi/10.1098/rspb.2009.0350 (visited on 11/26/2025).

A Program Code

main.py

```
import numpy as np
import PlotSummaryDataRobinson as psdr
import RobinsonCode as rc
import os
import yaml


def main():

    # Plot folder
    plot_folder = "plots"
    os.makedirs(plot_folder, exist_ok=True)

    # Read Parameters
    with open("params.yaml", "r", encoding="utf-8") as f:
        params = yaml.safe_load(f)

    qual_stddev_list = params["qual_stddev"] # quality noise
    threshold_stddev = params["threshold_stddev"] # threshold noise
    n = params["n"] # number of ants
    threshold_mean = params["threshold_mean"] # mean threshold

    colony_reps = params["colony_reps"] # colony repeats
    base_seed = params.get("base_seed", 42) # for reproducibility

    recruit_strength = params["recruit_strength"] # strength of recruitment
    quorum_k = params["quorum_k"] # quorum size

    quorum_sweep = params["quorum_sweep"] # quorum size to test in sweep plots
    qual_noise_fixed = params["qual_noise_fixed"] # fixed noise level for
        quorum sweep

    # Lists to store the average results for plotting
    acc_baseline = []
    acc_recruit = []
    acc_both = []
    time_baseline = []
    time_recruit = []
    time_both = []

    example_accepts = None
    example_accept_time = None

    # Loop through every noise level in list
    for qual_noise in qual_stddev_list:

        # No recruitment and no quorum
        temp_acc = []
        temp_time = []
        for rep in range(colony_reps):
            np.random.seed(base_seed + rep) # Change seed slightly each rep
            res = run_robinson(qual_noise, n, threshold_mean, threshold_stddev,
                0, 0.0)
            temp_acc.append(res[0]) # accuracy
            temp_time.append(res[1]) # time
        acc_baseline.append(np.mean(temp_acc))
        time_baseline.append(np.mean(temp_time))

        # Recruitment Only
```

```
        temp_acc = []
        temp_time = []
        for rep in range(colony_reps):
            np.random.seed(base_seed + 100 + rep) # Use a different seed range
            res = run_robinson(qual_noise, n, threshold_mean, threshold_stddev,
                0, recruit_strength)
            temp_acc.append(res[0])
            temp_time.append(res[1])
        acc_recruit.append(np.mean(temp_acc))
        time_recruit.append(np.mean(temp_time))

        # Recruit + Quorum
        temp_acc = []
        temp_time = []
        for rep in range(colony_reps):
            np.random.seed(base_seed + 200 + rep)
            res = run_robinson(qual_noise, n, threshold_mean, threshold_stddev,
                quorum_k, recruit_strength)

            # If the colony chose the good nest
            temp_acc.append(1.0 if res[4] == 2 else 0.0)

            # Colony decision time
            temp_time.append(res[5] if res[5] is not None else np.nan)

            # Save one representative run for plot 5
            if qual_noise == qual_noise_fixed and rep == 0:
                example_accepts = res[2]
                example_accept_time = res[3]

        # Nanmean in case some colonies never reached quorum
        acc_both.append(np.nanmean(temp_acc))
        time_both.append(np.nanmean(temp_time))

    # Package lists into dictionaries
    accuracy_results = {"baseline": acc_baseline, "recruitment only":
        acc_recruit, "recruit+quorum": acc_both}
    time_results = {"baseline": time_baseline, "recruitment only": time_recruit
        , "recruit+quorum": time_both}

    # Accuracy vs Noise Plot
    psdr.PlotAccuracyConditions(qual_stddev_list, accuracy_results,
                                os.path.join(plot_folder, "
                                    accuracy_vs_noise_conditions.png"), "
                                    Accuracy vs Noise")

    # Decision time vs Noise Plot
    psdr.PlotTimeConditions(qual_stddev_list, time_results,
                            os.path.join(plot_folder, "
                                decision_time_vs_noise_conditions.png"),
                            "Decision time vs Noise")

    # Quorum Sweep
    q_acc = []
    q_time = []

    for qk in quorum_sweep:
        colony_accs = []
        colony_times = []

        for rep in range(colony_reps):
            seed = base_seed + rep * 100 + qk
            np.random.seed(seed)
```

```
            # Run simulation with the current quorum size
            res_q = run_robinson(qual_noise_fixed, n, threshold_mean,
                threshold_stddev, qk, recruit_strength)
            colony_accs.append(1.0 if res_q[4] == 2 else 0.0)
            colony_times.append(float(res_q[5]) if res_q[5] is not None else np
                .nan)

        # Take if good nest pick and how long it took
        q_acc.append(float(np.nanmean(colony_accs)))
        q_time.append(float(np.nanmean(colony_times)))

    # Accuracy vs Quorum Plot
    psdr.PlotAccuracyVsQuorum(quorum_sweep, q_acc, os.path.join(plot_folder, "
        accuracy_vs_quorum.png"),
                              "Accuracy vs Quorum")

    # Decision Time vs Quorum Plot
    psdr.PlotTimeVsQuorum(quorum_sweep, q_time, os.path.join(plot_folder, "
        time_vs_quorum.png"), "Time vs Quorum")

    # If example data not None, plot the dynamics in Plot 5
    if example_accepts is not None:
        nest_labels = {1: "Poor nest", 2: "Good nest"}

        psdr.PlotCommitmentDynamics(example_accepts, example_accept_time,
            nest_labels,
                                    os.path.join(plot_folder, "
                                        commitment_dynamics_example.png"), "
                                        Commitment Dynamics")


def run_robinson(qual_val, n, threshold_mean, threshold_stddev, quorum_k=0,
    recruit_strength=0.0):

    probs = np.array([[0.91, 0.15, 0.03],
                      [0.06, 0.80, 0.06],
                      [0.03, 0.05, 0.91]])

    time_means = np.array([[1, 36, 143],
                           [36, 1, 116],
                           [143, 116, 1]])

    time_stddevs = time_means / 5

    quals = np.array([-np.inf, 4, 6])

    qual_stddev = np.array([0.0, qual_val, qual_val])

    accuracy, mean_decision_time, accepts, accept_time, colony_choice,
        colony_time, quorum_times = rc.RobinsonCode(
        n, quals, probs, threshold_mean, threshold_stddev,
        qual_stddev, time_means, time_stddevs,
        quorum_k=quorum_k,
        recruit_strength=recruit_strength
    )

    return accuracy, mean_decision_time, accepts, accept_time, colony_choice,
        colony_time, quorum_times


if __name__ == "__main__":
    main()
```

RobinsonCode.py

```
import numpy as np


def RobinsonCode(n, quals, probs, threshold_mean,
                 threshold_stddev, qual_stddev, time_means, time_stddevs,
                 quorum_k=0, recruit_strength=0.0):

    #   n = number of replicates (>=1)
    #   quals = row vector of m site qualities
    #           (quals(1) = home site
    #           quality: -Inf for no effect of home site quality on searching)
    #   discovery_probabilities = m * m matrix of discovery probabilities from
    #     column site to row site (N.B. columns should sum to 1)

    #   threshold_mean: mean population threshold for site acceptability
    #   threshold_stddev: standard deviation in population thresholds

    #   qual_stddev: standard deviation in quality assessments: **AOP**
    #   time_means: m * m matrix of mean travel times from column site to row
    #     site (N.B. should probably be symmetric)
    #   time_stddevs: m * m matrix of travel time standard deviations, from
    #     column site to row site (N.B. should probably be symmetric)

    # quorum_k = quorum size (0 means no quorum)
    # recruit_strength = strength of recruitment bias (0 means no recruitment)

    nestNum = probs.shape[0]  # number of sites

    # Final nest chosen by each ant
    accepts = np.zeros([n], dtype=int)

    # Time taken by each ant
    current_time = np.zeros([n])

    # discovery times and visit counts
    discovers = np.zeros([nestNum, n])
    visits = np.zeros([nestNum, n])


    Ants = []
    for i in range(n):
        ant = {'path': [],
               't': [],
               'thresh': 0,
               'selected': 0}
        Ants.append(ant)

    # Time each ant accepts a nest
    accept_time = np.full(n, np.nan)

    # Number of ants accepted each nest
    recruiters = np.zeros(nestNum, dtype=int)

    # Set the maximum number of steps for each ant
    Max_num_steps = 1000

    for i in range(n):

        # Monte Carlo simulation of one ant
        # this sets up the output variables **AOP**

```

```
        # This holds the time it has taken before an ant has made the first
        # recruitment **AOP**
        current_time[i] = 0

        # this is a variable which holds where ant i currently is **AOP**
        accepts[i] = 0  # ant starts in home site

        # this is a matrix which holds the time at which the ant discovers
        # sites 1 to N where N= number of sites;
        # As ant is currently in the home site it never 'discovers' it
        # so set this (arbitrarily) to -1 **AOP**
        discovers[0,i] = -1 # ant is already in home site

        # this is a matrix which holds the number of times an ant visits
        # sites 1 to number of sites. As it is already in the home site the
        # 1st element is set to 1 **AOP**
        visits[0,i] = 1 # ant is already in home site

        # initialise the variables for the other home sites. it hasn't been to
        # any of the others so the time to 1st discovery is 0 and the number
        # of visits = 0  **AOP**
        for j in range(1, nestNum): # could be a problem here?

            discovers[j, i] = 0 # ant has not discovered or visited other sites
            visits[j, i] = 0

        # sample and set the ant's acceptance threshold **AOP**
        thresh = threshold_stddev * np.random.randn() + threshold_mean

        # set up some output variables *** AOP
        num_step = 0
        Ants[i]['path'].append(accepts[i])
        Ants[i]['t'].append(current_time[i])
        Ants[i]['thresh'] = thresh
        Ants[i]['selected'] = 0

        # this is now the main loop of the program. Essentially it says:
        # 1. for the current site, check to see if the ant accepts it based on
        #    it's threshold and a randomly selected quality based on the
            quality
        #    of the site **AOP**
        # 2. Do this until a site is accepted
        while Ants[i]['selected'] == 0:

            # check the quality of the current nest
            perceivedQuality = qual_stddev[accepts[i]] * np.random.randn() +
                quals[accepts[i]]

            # if the perceived nest quality is above the threshold, select it
            if perceivedQuality >= Ants[i]['thresh']:
                Ants[i]['selected'] = 1

                # Record acceptance time for quorum calculations
                accept_time[i] = current_time[i]

                # Update recruitment count for the chosen nest
                recruiters[accepts[i]] += 1
                break

            # if you have exceeded the max number of steps without stopping
            # break out of the algorithm
            if num_step > Max_num_steps:
                Ants[i]['selected'] = 0
```

```
                accept_time[i] = np.nan
                break

            # movement probabilities from current site
            p = probs[:, accepts[i]].astype(float).copy()

            # Recruitment bias toward nests with more accepted ants
            # Positive feedback mechanism
            if recruit_strength > 0.0:

                bias = recruit_strength * recruiters.astype(float)
                bias[0] = 0.0  # no recruitment to home
                p = p + bias

                # Normalise
                p = np.clip(p, 0.0, None)
                s = p.sum()
                if s > 0:
                    p = p / s
                else:
                    # Fall back to the original transition probs
                    p = probs[:, accepts[i]].astype(float).copy()

            # probablistically pick one of the new sites to go to
            # unifrnd(0,1) generates a uniformly distributed number
            # between 0 and 1
            ran = np.random.uniform()
            newsite = 0
            while ran > p[newsite]:
                ran = ran - p[newsite]
                newsite = newsite + 1

            # update the time taken with normally-distributed time-step size
            # (>=1) **AOP**
            delta = max(1, time_stddevs[newsite, accepts[i]] * np.random.randn
                () + time_means[newsite, accepts[i]])
            current_time[i] = current_time[i] + delta

            # update ant's current site, accepts, **AOP**
            # discovers, and the number of times it has been visited, visits **
                AOP**
            accepts[i] = newsite

            # if it hasn't discovered this site before, update the time that it
            # 1st discovered it, in discovers **AOP**
            if discovers[newsite, i] == 0:
                discovers[newsite, i] = current_time[i]

            # update the number of times it has visited this site **AOP**
            visits[newsite, i] = visits[newsite, i] + 1

            # Update the output variables **AOP**
            num_step = num_step + 1
            Ants[i]['path'].append(accepts[i])
            Ants[i]['t'].append(current_time[i])

        # record number of steps taken
        Ants[i]['numSteps'] = num_step

    # Calculate mean decision time
    mean_decision_time = float(np.mean(current_time))

    # Calculate accuracy by ants chose the good nest
```

```
    accuracy = np.sum(accepts == 2) / len(accepts)

    # Which nest reaches quorum first, None if no quorum reached
    colony_choice = None

    # Time at which the winner nest hits quorum
    # None if no quorum reached
    colony_decision_time = None

    # Time when nest j reaches quorum
    quorum_times = np.full([nestNum], np.inf)

    # Quorum
    if quorum_k > 0:
        best_time = 999999999

        for nest_idx in range(1, nestNum): # Start from 1 to skip home

            # Manually find which ants chose this nest
            nest_times = []
            for i in range(n):
                if accepts[i] == nest_idx:
                    # Check if they actually accepted
                    if not np.isnan(accept_time[i]):
                        nest_times.append(accept_time[i])

            # Check if this nest even reached the quorum size
            if len(nest_times) >= quorum_k:
                # Sort them to find when the ant arrive
                nest_times.sort()
                this_quorum_time = nest_times[quorum_k - 1]

                # Update the wiinner
                if this_quorum_time < best_time:
                    best_time = this_quorum_time
                    colony_choice = nest_idx
                    colony_decision_time = float(best_time)

        # Fill quorum_times for the return variable if needed
        for nest_idx in range(1, nestNum):
            nest_times = [accept_time[i] for i in range(n) if accepts[i] ==
                nest_idx and not np.isnan(accept_time[i])]
            nest_times.sort()
            if len(nest_times) >= quorum_k:
                quorum_times[nest_idx] = nest_times[quorum_k - 1]

    # Take returns of accuracy, mean_decision_time, accepts,
    # accept_time, colony_choice, colony_decision_time, quorum_times
    return accuracy, mean_decision_time, accepts, accept_time, colony_choice,
        colony_decision_time, quorum_times
```

PlotSummaryDataRobinson.py

```
import matplotlib.pyplot as plt
import numpy as np

# Function to plot how accurate the ants and colony are as noise increases
def PlotAccuracyConditions(x_vals, series_dict, out_path, title):

    # Create figure
    plt.figure(figsize=(8, 5))

```

```
    # Loop through each condition
    for label, y_vals in series_dict.items():
        plt.plot(x_vals, y_vals, marker='o', label=label)

    plt.xlabel("Nest Quality Noise (sigma)") # label x-axis
    plt.ylabel("Accuracy")  # label y-axis
    plt.title(title) # set title
    plt.grid(True)  # add grid
    plt.legend() # add legend
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()

# Function to plot how long it takes to make a decision as noise increases
def PlotTimeConditions(x_vals, series_dict, out_path, title):

    # Create figure
    plt.figure(figsize=(8, 5))

    # Loop through each condition
    for label, y_vals in series_dict.items():
        plt.plot(x_vals, y_vals, marker='o', label=label)

    plt.xlabel("Nest Quality Noise (sigma)") # label x-axis
    plt.ylabel("Decision Time") # label y-axis
    plt.title(title) # set title
    plt.grid(True) # add grid
    plt.legend() # add legend
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()

# Function to plot to observe if accuracy changes as increasing the quorum size
def PlotAccuracyVsQuorum(q_vals, acc_vals, out_path, title):

    # Create figure
    plt.figure(figsize=(8, 5))

    # Plot accuracy as a function of quorum size
    plt.plot(q_vals, acc_vals, marker='o')

    plt.xlabel("Quorum size (Q)") # label x-axis
    plt.ylabel("Accuracy") # label y-axis
    plt.title(title) # set title
    plt.grid(True) # add grid
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()

# Function to plot to see if takes longer to reach a decision with a higher
    quorum
def PlotTimeVsQuorum(q_vals, time_vals, out_path, title):

    # Create figure
    plt.figure(figsize=(8, 5))

    # Plot commitment time as a function of quorum size
    plt.plot(q_vals, time_vals, marker='o')

    plt.xlabel("Quorum size (Q)") # label x-axis
    plt.ylabel("Time to quorum") # label y-axis
    plt.title(title) # set title
    plt.grid(True) # add grid
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()

```

```python
# Function to plot to see how fast the colony builds up at each nest until they hit the quorum threshold
def PlotCommitmentDynamics(accepts, accept_time, nest_labels, out_path, title):
    plt.figure(figsize=(8, 5))

    # Filter any ants that did not choose a nest
    ok_times = accept_time[~np.isnan(accept_time)]
    if len(ok_times) == 0:
        plt.close()
        return

    # Create the x-axis for the plot
    max_t = np.max(ok_times)
    t_steps = np.linspace(0, max_t, 200)

    # Loop through each nest and count arrivals
    for nest_id in sorted(set(accepts)):
        if nest_id == 0:
            continue  # skip home site

        # Get times for just this nest
        this_nest_times = accept_time[accepts == nest_id]
        this_nest_times = this_nest_times[~np.isnan(this_nest_times)]
        this_nest_times.sort()  # sort them

        # Calculate how many ants are there at each time step
        ant_counts = np.searchsorted(this_nest_times, t_steps, side='right')
        plt.plot(t_steps, ant_counts, label=nest_labels.get(nest_id, f"Nest {nest_id}"))

    plt.xlabel("Time") # label x-axis
    plt.ylabel("Number of ants committed") # label y-axis
    plt.title(title) # set title
    plt.grid(True) # add grid
    plt.legend()
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()
```

params.yaml

```yaml
qual_stddev: [0.1, 0.5, 1.0, 1.5, 3.0, 4.0] # Nest quality noise
threshold_stddev: 0.6 # Threshold noise
n: 500 # number of ants
threshold_mean: 4.5 # threshold
colony_reps: 30 # number of colony
base_seed: 42 # for reproducibility
recruit_strength: 0.05 # strength of recruitment
quorum_k: 20  # quorum size
quorum_sweep: [5, 10, 15, 20, 30, 40, 60, 80] # quorum sizes tested in sweep
qual_noise_fixed: 1.5 # fixed noise level used in quorum sweep plots
```