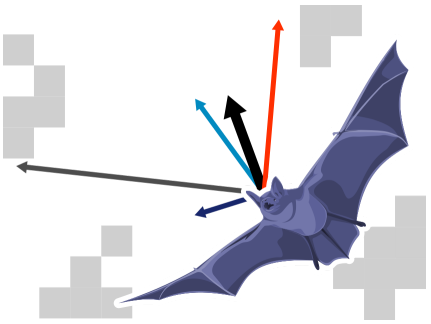
ODD protocol for a bat sensory network formation model



This is an ODD protocol document ("Overview, Design concepts, and Details"), which provides a detailed description of our model presented in:

Roeleke, M., Schlägel, U. E., Gallagher, C. A., Pufelski, J., Blohm, T., Nathan, R., Toledo, S., Jeltsch, F., & C. C. Voigt. Insectivorous bats form mobile sensory networks to optimize prey localization. (Submitted.)

1. Purpose and patterns

Searching for food patches within a sensory network may allow for animals to more efficiently locate patchily distributed and ephemeral resources, however there exists little knowledge on the potential benefits of this strategy and the conditions under which it is advantageous. The purpose of the model is to assess if and under which conditions sensory networking of insectivorous bats can lead to observed differences in the amount of time it takes bats to find food in varying environments.

When developing the model we used patterns elucidated in the empirical portion of the manuscript (Fig. 4c & d), which document changes in distance to conspecifics associated with varying initial distances for model calibration (see ODD Element 7.8). Three additional movement patterns (time to first forage, distance from roost to forage cell, and straightness index) were used to evaluate the model (ODD Element 7.8).

2. Entities, state variables, and scales

The model comprises two entities: **landscape cells** and individual **bat agents**.

Landscape cells are square grid cells characterized by their position and whether they contain prey resources (*food*). If they do contain food, then cells are additionally characterized by the ID of the patch, defined as clusters of cells with food, they belong to (*patch-ID*) and whether they have been found by bat (*found*) (Table S2.1). Here food represents aggregations of insects forming static swarms which may persist in a location for some time. These could represent mating aggregations, emergence events, or groups driven by climatic conditions. The use of static food patches for the

short timescales covered in the model (minutes to few hours) was supported by observations in the tracking data where individuals exhibited area-restricted search behavior in one spot for several minutes (data not shown). For simplicity, prey resources were expressed as a property of landscape cells (i.e., food present or not) rather than being explicitly described.

Each grid cell covers 75x75 m. This cell size was selected as bats in the empirical tracking dataset tended to keep approximately this distance from each other when hunting (data not shown). The total spatial extent of the landscape is 80x80 square grid cells, covering an area of 6000x6000 m. This landscape size was selected to match the extent of the empirically studied foraging area. The model landscape has closed boundaries, i.e. was bounded at the extents and not toroidal. The food cell distribution is controlled by the parameter *no-patch*, which sets the total number of patches, or spatial aggregation level, of the landscape. This value can range between 1 and the number of food cells used (*no-food-cells*), leading to spatial aggregation levels of 100 - 0%, respectively.

Bat agents are characterized by 17 variables related to their movement behavior and interactions with other bats (Table S2.1). Bats move through the environment using four vector-based movement behaviors: a random walk process, and three processes which are influenced by interactions with other bats (*attraction*, *alignment*, and *avoidance*). Each vector contains a direction and strength component. Directions point to a local x and y position (*attract-vect, align-vect, avoid-vect,* and *rw-vect*), while strengths, or weights, range between zero to one (*attract-strength, align-strength, avoid-strength,* and *rw-strength*), where one is the strongest possible value (see ODD Element 7 for details). When bats have a conspecific (*conspecific*), their resulting movement direction (*turn-angle-real*) is determined from the interacting movement processes (Fig. S2.1). Bats without conspecifics only use the random walk behavior. Their movement speed (*step-length*) is pulled from a distribution consistent with steps taken by bats in the empirical tracking data.

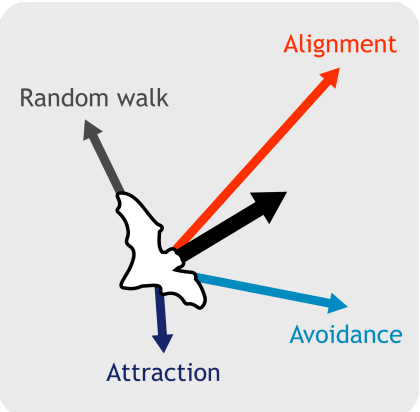


Figure S2.1. *The four behaviors which drive bat movement direction. The Attraction, Alignment, and Avoidance vectors are determined based on the distance to and foraging behavior of a conspecific, if present, while the Random walk behavior occurs for all bats regardless of whether they have a conspecific. Arrow length represents the relative strength, or weight, of each vector. Each vector is independently assessed and the resulting direction is taken as a mixture of all four movement vectors (black).*

Bats search for food as they move in the landscape. When they find a food cell (*food-found* = "true"), they record the food's location (*food-cell*) and fly in the vicinity of that cell for the remainder of the simulation period. If a bat encounters an occupied food cell in a patch which contains food cells that are unoccupied, it can target an unoccupied cell using the variables *targ-cell* and *flying-towards-food* (see *food-check* section for details)

Time is modelled using discrete time steps, where each step represents 8 seconds. This interval was used as it matched the sampling interval of the tracking data in the empirical portion of the study. Model runs continue until all bats have found an available food cell, generally within the span of minutes to several hours depending on the number of bats and landscape values used.

Table. S2.1. *State variables used in the model. All landscape cells or global state variables are static, while all bat state variables are dynamic. Maximum values for step lengths came from the empirical tracking data, while maximum values for the two timers (*hunting conspecific timer and move away timer*) were set to a maximum of ~ five minutes (40 time steps).*

State variable	***Code***	**Description [unit]**	**Type**	**Range**
Landscape cells				
Number of patches	*no-patch*	Number of food cell aggregations (e.g. patches) [unitless]	Integer	[1.. *no-food-cells*]
Whether food present	*food*	Whether or not food is found in the cell [unitless]	Boolean	"true" or "false"
Whether food has been found	*found*	Whether the cell has been found by a bat [unitless]	Boolean	"true" or "false"
Patch ID	*patch-id*	ID of patch that the food cell is a part of [unitless]	Integer	[1..*no-patch*]
Bats				
X, Y	*xcor, ycor*	Position [cells]	Float	-
Step length	*step-length*	Flying speed [cells per time step]	Float	[0,3.4]
Turning angle	*turn-angle-real*	Actual turning angle of bats after all movement processes have occurred	Float	[-180,180]

		[degrees]		
Attraction vector	*attract-vect*	Movement vector directing bat to approach conspecific [cells]	List of local positions	-
Attraction strength	*attract-strength*	Relative weighting of attraction vector. Used in determining turning angle [unitless]	Float	[0,1]
Alignment vector	*align-vect*	Movement vector directing bat to maintain distance to conspecific [cells]	List of local positions	-
Alignment strength	*align-strength*	Relative weighting of alignment vector. Used in determining turning angle [unitless]	Float	[0,1]
Avoidance vector	*avoid-vect*	Movement vector directing bat to move away from conspecific [cells]	List of local positions	-
Avoidance strength	*avoid-strength*	Relative weighting of avoidance vector. Used in determining turning angle [unitless]	Float	[0,1]
Random walk vector	*rw-vect*	Movement vector directing bat to randomly walk in landscape [cells]	List of local positions	-
Random walk strength	*rw-strength*	Relative weighting of the random walk vector. Used in determining turning angle [unitless]	Float	[0,1]
Conspecific	*conspecific*	Nearest bat to focal bat. Used to determine movement behavior [unitless]	String	all bat IDs or “nobody”
Food found	*food-found*	Whether or not a bat has found food. The model proceeds until all bats have found food [unitless]	Boolean	“true” or “false”
Food cell	*food-cell*	Cell where food was found. Used to perform hunting bat movements [unitless]	Cell	All cells
Hunting conspecific timer	*hunting-consp-ticker*	Timer to keep track of how long bats have been networking with a conspecific that is hunting. Used to allow bats to leave conspecifics which have already found food to continue sampling the landscape [time steps]	Integer	[0..40]
Move away timer	*move-away-ticker*	Timer to keep track of how long bats network only with searching bats after leaving a hunting conspecific. Used to allow bats to leave conspecifics which have already found food to continue sampling the landscape [time steps]	Integer	[0..40]
Target cell	*targ-cell*	Food cell that bat targets when it has encountered an occupied food cell in a patch which contains additional unoccupied food cells. This allows bats to travel through food patches to find unoccupied food cells	Cell	All cells

		once they have encountered a patch [unitless]		
Flying towards food	*flying-towards-food*	Whether bat has identified and is currently flying towards a target cell (*targ-cell*), this overrides all other movement behaviors [unitless]	Boolean	"true" or "false"

3. Process overview and scheduling

***Processes*:** To simulate the movements of interacting bats, the model proceeds using seven key processes: one related to selecting neighboring bats as conspecifics (*set-conspecific*), five related to calculating movement behavior (*attraction*, *alignment*, *avoidance*, *random-walk*, *hunting-fly*, and *bats-move*), and one process for identifying if a food cell has been found (*food-check*). The submodels for each process are described in detail in ODD Element 7.

Bats proceed through these processes once per time step, but only run *attraction*, *alignment*, and *avoidance* procedures if they have not found food and have identified a conspecific in the *set-conspecific* step. Bats who have found food run the *hunting-fly* procedure once per time step.

***Schedule*:** Bats in the model update their movements once per time step. Procedures all occur in the same predetermined order (Figure S3.1). All bats which have not yet found food first check for neighboring bats using the *set-conspecific* procedure. Then bats pull their step length from a random gamma distribution based on their foraging status (details in ODD Element 4.9). All bats which have a conspecific are first asked in a random order to calculate their *attraction*, *alignment*, *avoidance*, and *random-walk* vectors, then, in a separate step, all bats (again in a random order) are asked to move. These steps are executed separately so that bats are all on the same schedule, i.e., all calculate their movement direction, then all move (i.e., synchronous updating of their positions). Bats which have not found food and have no conspecific skip the *attraction*, *alignment*, and *avoidance* procedures and only run *random-walk* to determine their movement direction.

Once bats have calculated their movement vectors, they then determine their resulting direction in the *bats-move* procedure (based on the direction and strengths of each movement vector) and move. Moving and searching for food are broken up into three steps to avoid bats passing through but missing a food cell.

Bats which have found food do not check for conspecifics nor do they calculate any of the movement processes. They instead only fly around the cell they have found food in using the *hunting-fly* procedure.

The model proceeds until all bats have found food.

If producing outputs for calibration or evaluation, the *movement-outputs* procedure is executed at the beginning of the time step either once per 4 time steps (for calibration) or once every time step (for evaluation). For scenario outputs, once per every 8 time steps (~ a minute) observer outputs are calculated at the end of the time step using the *collect-outputs* procedure.

```
procedure Setup

  Generate landscape
  Initialize globals
  Generate bats

end procedure                                   Setup
```

```
procedure Go

  if leave-roost-tick < timestep and food-found = "false" (bats
  have left the roost but have not found food yet) then
  | set-conspecific (if any)
  | check-consp-foraging (check conspecific's foraging status)
  end if

  if food-found = “true” (bat is foraging) then
  | pull step-length randomly from a gamma distribution
  | based on empirical data from foraging bats
  else
  | pull step-length randomly from a gamma distribution
  | based on empirical data from nonforaging bats
  end if

  if food-found = “false” (bat is not foraging) then
  | if conspecific != “nobody” (they have a conspecific) then
  | | set vector ranges based on conspecific foraging state
  | | calculate attraction vector
  | | calculate alignment vector
  | | calculate avoidance vector
  | else
  | | set attraction, alignment, and avoidance vectors to 0
  | end if
  | calculate random-walk vector
  end if

  all bats update heading and fly

end procedure                          Every time step until
                                       all bats have found food
```

Figure S3.1. *Pseudo-code outlining the order of model process execution. Italicized text denotes naming used in code. All processes occurring in the dark grey box occur for each bat during each time step until all bats have found food.*

4. Design concepts

4.1. Basic principles

The model is built on the underlying theory that bats form sensory networks which may help them find food patches, as first proposed by Cvikel et al. (2015). The empirical portion of this study identified patterns in the movement behavior of neighboring insectivorous common noctule bats (*Nyctalus noctula*) which indicate the formation of sensory networks. The model was developed to quantify a potential benefit of these networks, a reduction in the length of time it takes bats to find food.

At the agent level, the model is rooted in Boids movement dynamics (Reynolds, 1987), where bats interact using three movement processes: attraction (bats are attracted towards conspecifics), alignment (bats maintain distance from conspecifics), and avoidance (bats turn away from conspecifics). Each movement process is represented as a separate submodel (ODD Element 7). This model differs from standard Boids models in that behaviors are based only on the closest conspecific rather than all other bats within a certain radius. This was done to be able to directly use the patterns established in the empirical portion of this study (main text Fig. 4c & d) for calibrating the movement processes, where only the nearest tagged conspecific was registered.

4.2. Emergence

The movement behavior and networking of bats emerge from bat movement decisions. The time it takes to find a food cell emerges from the landscape structure (number and placement of patches) and interactions between bats.

4.3. Adaptation

Bats adapt their behavior through adjustments to their movement direction: bats base their direction on the distance to their conspecific (if any) and the resulting weighted headings given by each of the four movement processes. The resulting behavior is determined primarily by the movement process which is the strongest at the current distance to the conspecific. The strength of each movement process at varying distances to the conspecific was calibrated using the empirical patterns (see ODD Element 7.8 for details).

4.4. Sensing

Bats can sense their location and the location, movement direction, and foraging behavior of other bat agents in their vicinity (240m). They can also sense whether food is present in cells within a 15m radius of their position (*prey-detection-range*).

4.5. Interaction

Bats directly interact with one another by influencing the movement direction of other bats. Bats only interact with one conspecific at a time in the model, forming dyads (see *set-conspecific* submodel below for details on how conspecifics are selected).

In nature, bats can use echolocation calls from conspecifics to distinguish the foraging behavior of other bats and eavesdrop on local prey availability (Gager, 2019; Kalko, 1995). In the model we assume that bats can detect and distinguish the foraging behavior of conspecifics at distances of 240 m. Though this distance is 1.5 times the presumed maximum conspecific detection distance of 160 m for common noctule bats (Voigt et al., 2021), this range of 240 m was used to match the statistical analyses presented in the main text which were used for model calibration.

Mediating interactions occur in the model through competition for food resources. Each food cell is available to only a single bat and once a food cell is occupied it becomes unavailable to all other bats for the duration of the simulation. This was done to ensure that bats could forage in food cells with sizes corresponding to the distances maintained between hunting bats in the empirical dataset. Though in nature multiple bats may be observed to forage simultaneously within the same 75x75 m area, hunting bats likely maintain distance to other bats due to factors not considered here, including acoustic jamming (Amichai et al., 2015), prey density, and competition. For this purpose we have limited one bat per food cell.

Bats which have found food cells continue to fly around in the vicinity of the food cell (to simulate hunting activity) and continue to influence the movement behavior of other searching bats.

4.6. Stochasticity

Bats leave the roost on a random time step between the start of the simulation and time step 37 (*leave-roost-tick*). This parameter staggers the emergence of bats from the roost so that all bats do not leave at the exact same time step.

The step lengths (i.e., flying speeds) of bat agents are randomly pulled once per time step from a gamma distribution fit to the step lengths of tracked bats in the empirical

dataset. Step length is based on foraging activity, such that bats pull from gamma distributions fit to bats which were exhibiting their current foraging behavior (i.e., hunting or searching). The maximum step length (3.4 cells $tick^{-1}$) was set as 10% higher than the maximum empirical step length recorded (3.042 cells $tick^{-1}$). Turning angles of bats in the random walk process (ODD Element 7.4.4) were also determined in this manner.

Food patches are placed randomly within the foraging zone (in radius *foraging-radius* of the landscape center). Each map generation results in a unique landscape containing the specified number of patches (*no-patch*).

4.7. Observation

Graphical output on the model interface shows the number of bats which have found food, the percentage of searching bats with a conspecific, the average network size, and the time it took each bat to find a food cell.

A subset of bats can be tracked for calibrating and evaluating movement processes. For these tracked bats, the distance to their conspecific in the previous and current recorded time step (32 sec time intervals were used for calibration), difference in distances between the two recordings, and x and y positions are all observed.

Network sizes were collected by assessing the number of bats which occur in a single network in a time step. Though bats can only have one conspecific at a time, larger networks can emerge from bats which have a different conspecific than the bat which sees them as its conspecific (e.g., bat 1 can see bat 2 as its conspecific, while bat 2 sees bat 3 as its conspecific, resulting in a network size of 3).

For scenarios, the time it took all bats to find food, for 95% of bats to find food, and for each bat to find food (adjusted for the time step at which it left the roost) were recorded.

5. Initialization

Upon initialization, a number of simulated bats are created based on the value of the input parameter *n-bats*. All bats are identical and are generated with all state variables initialized at 0, “false“, or “nobody”, where relevant. The one exception is the *leave-roost-tick* parameter which is selected randomly with a maximum number corresponding to 37 time steps (~5 minutes).

Bats are generated at the location of the roost (coordinate = (0, -31.33)), which was based on the roost position in the real foraging area in the Uckermark site (Fig. S5.1). Bats do not start forming networks until after leaving the roost.

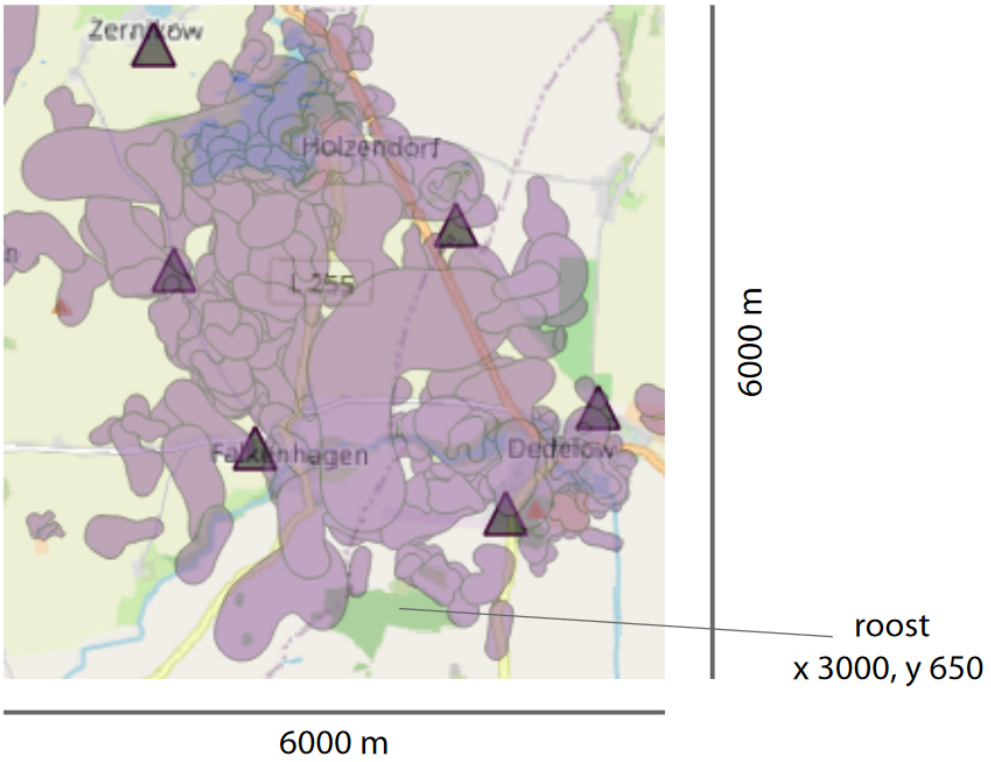


Figure S5.1. *Study area in the Uckermark region that the model landscape was based on, denoting location of the roost. Foraging patches (95% isopleths of kernel density estimates based on localizations during hunting) for 75 bats shown in purple.*

Food cells are placed randomly within the foraging area using the *setup-maps* submodel. Landscapes can vary in their number and location of patches (Figure S5.2). The number of food cells used (213) was selected as the mean number of 75x75 m cells occupied by bats in home range maps in the empirical dataset for days where a minimum of five bats were tracked concurrently. Landscapes are generated as follows: all patches are initialized with their *food* and *found* variables (Table S2.1) set to “false“ and their *patch-id* set to 0. The roost is then created and the first cell for each patch is then placed in the landscape at a minimum distance of 2 cells from other food cells. Then remaining food cells are placed one at a time in each patch, until all food cells are placed in the landscape. Each food cell must neighbor a cell in its patch but also not touch a food cell from another patch on any of its eight sides, ensuring that food patches do not touch.

Simulation experiments are identical in setup beyond varying parameter values, e.g., running with 160 versus 80 simulated bats.

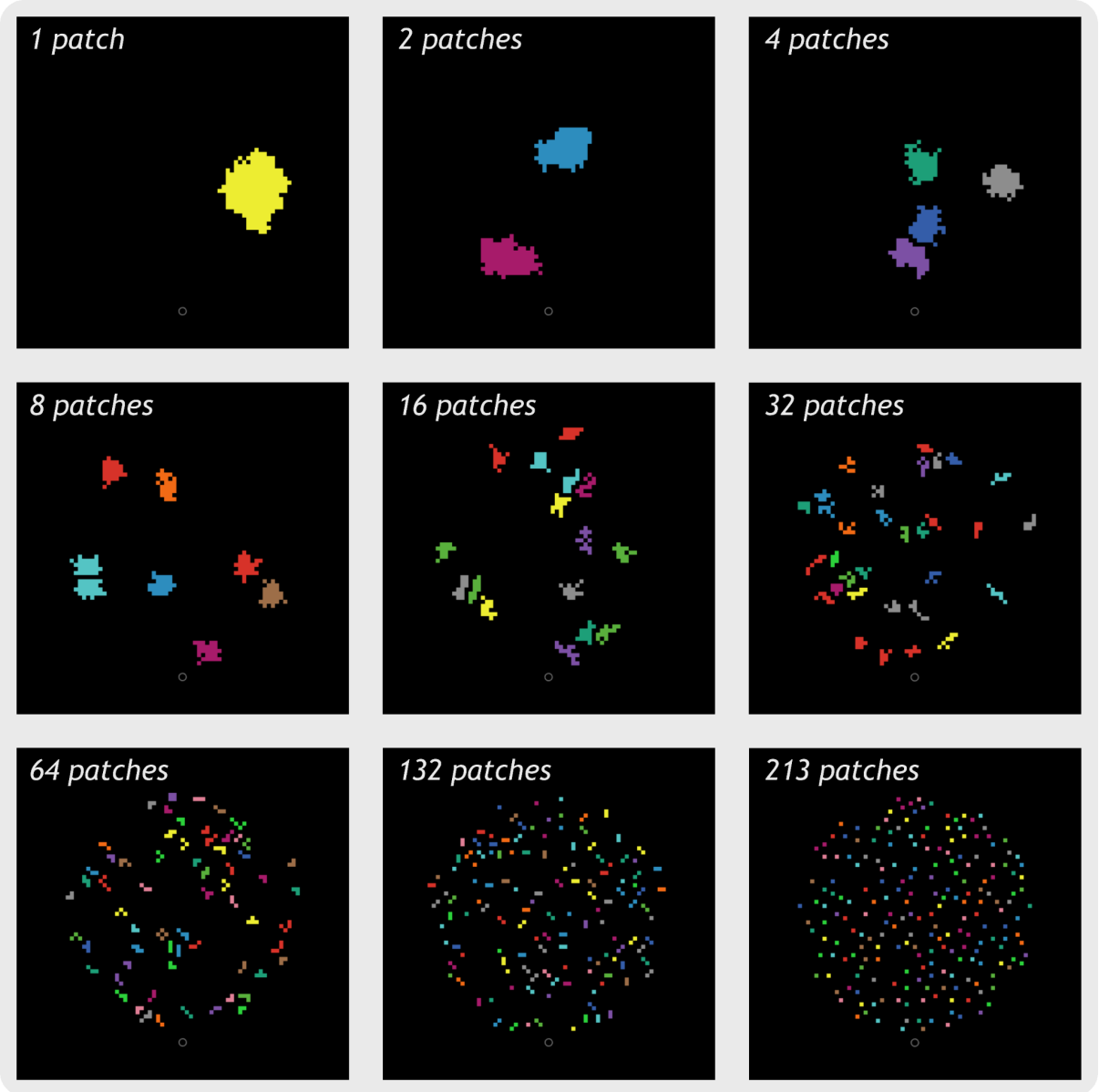


Figure S5.2. *Example landscapes generated with differing numbers of patches.*

6. Input data

The model does not use input data to represent time-varying processes.

7. Submodels

Table. S7.1. *Model parameter definitions and values.*

Parameter	***Code***	**Description [unit]**	**Source**	**Value**
Number of food cells	*no-food-cells*	Number of food cells in the environment [cells]	Based on empirical data	213
Attraction range	*attraction-range*	Range where attraction occurs [m]	Based on empirical patterns	240
Alignment range - Searching	*alignment-range-s*	Range where alignment is at its maximum for bats with a searching conspecific [m]	Calibrated	60
Alignment range - Hunting	*alignment-range-hunt*	Range where alignment is at its maximum for bats with a hunting conspecific [m]	Calibrated	120
Avoidance range	*avoidance-range*	Range where avoidance is at its maximum [m]	Based on empirical patterns	0.01
Attraction modifier - Searching	*attract-mod-s*	Attraction vector strength modifier for bats with a searching conspecific [unitless]	Calibrated	0.006
Attraction modifier - Hunting	*attract-mod-hunt*	Attraction vector strength modifier for bats with a hunting conspecific [unitless]	Calibrated	0.0005
Alignment modifier - Searching	*align-mod-search*	Alignment vector strength modifier for bats with a searching conspecific [unitless]	Calibrated	0.02
Alignment modifier - Hunting	*align-mod-hunt*	Alignment vector strength modifier for bats with a hunting conspecific [unitless]	Calibrated	3
Avoidance modifier - Searching	*avoid-mod-search*	Avoidance vector strength modifier for bats with a searching conspecific [unitless]	Calibrated	0.1
Avoidance modifier - Hunting	*avoid-mod-hunt*	Avoidance vector strength modifier for bats with a hunting conspecific [unitless]	Calibrated	0
Random walk modifier - Searching	*rw-mod-search*	Random walk vector strength modifier for bats with a searching conspecific or no conspecific [unitless]	Calibrated	0.4
Random walk modifier - Hunting	*rw-mod-hunt*	Random walk vector strength modifier for bats with a hunting conspecific [unitless]	Calibrated	0
Leave roost tick	*leave-roost-tick*	Time step where a bat will leave the roost [time steps]	Arbitrarily selected	[1..37]
Prey detection	*prey-detection-range*	Radius where prey patches can be	Boonman et al.,	15

range		detected [m]	2019	
Radius of the foraging area	*foraging-radius*	Radius of the foraging area from the landscape center [cells]	Based on empirical landscape	35
Turning angle alpha - Searching	*turn-ang-alpha-search*	Turning angle gamma distribution alpha parameter for bats which are searching [unitless]	From empirical dataset	0.66
Turning angle lambda - Searching	*turn-ang-lambda-search*	Turning angle gamma distribution lambda parameter for bats which are searching [unitless]	From empirical dataset	2.62
Turning angle alpha - Hunting	*turn-ang-alpha-hunt*	Turning angle gamma distribution alpha parameter for bats which are hunting [unitless]	From empirical dataset	0.78
Turning angle lambda - Hunting	*turn-ang-lambda-hunt*	Turning angle gamma distribution lambda parameter for bats which are hunting [unitless]	From empirical dataset	0.98
Step length alpha - Searching	*step-len-alpha-search*	Step length gamma distribution alpha parameter for bats which are searching [unitless]	From empirical dataset	3.4
Step length lambda - Searching	*step-len-lambda-search*	Step length gamma distribution lambda parameter for bats which are searching [unitless]	From empirical dataset	5.6
Step length alpha - Hunting	*step-len-alpha-hunt*	Step length gamma distribution alpha parameter for bats which are hunting [unitless]	From empirical dataset	1.5
Step length lambda - Hunting	*step-len-lambda-hunt*	Step length gamma distribution lambda parameter for bats which are hunting [unitless]	From empirical dataset	6.7

7.1. Check for conspecific (set-conspecific)

At the beginning of each time step searching bats select their conspecifics (if any) (Fig. S7.1). Bats first check if there are any other bats within the maximum range (240m). If so, all bats within this region are saved to a temporary list called *conspecifics*. If there are no bats within this region, then the bat sets its conspecific to "nobody" and proceeds to the *random-walk* submodel (Fig S7.1 left). Bats which have been networking with hunting conspecifics for too long will ignore conspecifics that are currently hunting when selecting their conspecific (see submodel *check-consp-hunting* below for details). To do this, bats which have identified any possible conspecifics in their area first check if they are currently moving away from hunting bats (i.e., *move-away-ticker* > 0). If so, then the bat looks for the closest

searching bat in the conspecifics list and sets it as its conspecific (Fig S7.1 center). If there are no searching bats in the list, then the conspecific is set to “nobody”. If the conspecifics list is not empty and the bat has a *move-away-ticker* value of 0 (i.e., not moving away from hunting bats), then it chooses the closest bat in its radius as its conspecific (regardless of foraging behavior of the other bat)(Fig S7.1 right).

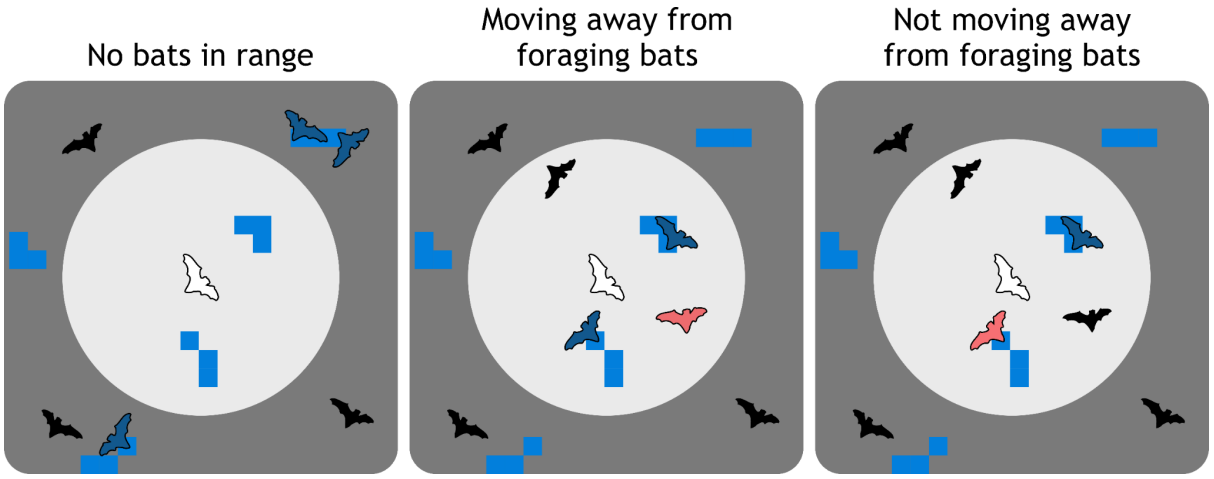


Figure S7.1. *Schematic illustrating the conspecific selection process of bats depending on whether there are any neighboring bats and the state of the focal bat (i.e., whether it is currently moving away from other bats). The focal bat is shown in white and the attraction range visualized using a light grey circle. Blue squares represent food cells and blue bats are bats which are currently hunting. The conspecific, if any, is shown in pink. Bats keep track of how long they have had a conspecific that is hunting using the hunting-consp-ticker parameter and after this timer reaches five minutes (40 time steps), bats ignore hunting conspecifics for an additional five minutes (center panel).*

7.2. Check if current conspecific is hunting (check-consp-hunting)

In the *check-consp-hunting* procedure bats keep track of how long they have had a conspecific which has been hunting. This procedure is run as a safety net for bats to disengage from hunting bats to avoid getting stuck in networks with bats which may be fixed in a location that does not contain any empty food cells. Flying around a hunting bat can help bats find empty food cells nearby when present, particularly when the patch size is large, but can also be a distraction to searching bats when no available food cells are nearby. In the model, bats will stay with a hunting conspecific for 40 time steps (~ 5 minutes), using a parameter called *hunting-consp-ticker* to keep track. After this timer reaches 40 time steps, bats then ignore hunting bats when selecting their conspecifics again for 40 time steps, keeping track using the parameter

move-away-ticker. *Move-away-ticker* is used in the *set-conspecific* submodel (above) when selecting a conspecific from neighboring bats.

7.3. Set ranges used (set-ranges)

This submodel is used to convert the ranges for each of the three Boids-based movement processes from distance in meters to distance in units of cell size. Additionally, as the calibration resulted in different values used for the alignment range for bats depending on whether they have a hunting or searching conspecific, this process is also used to select the correct range to use and convert that to distance in cells.

7.4. Movement processes:

The following four submodels all relate to the movement behavior of modelled bats. Attraction, alignment, and avoidance are carried out only by bats which have a conspecific, while the random walk submodel is executed by all searching bat agents. All submodels consist of two parts: first a vector is drawn that indicates the heading identified by each process, then the strength (i.e., weight) of the vector is calculated. The directions and strengths of all vectors are taken together to determine the final heading of the bat in the time step.

7.4.1. Attraction

Bats which are attracted to other bats orient their heading in the direction of their conspecific. They do this by first pulling the x and y positions of their conspecific as temporary variables. The focal bat then draws the attraction vector in local space spanning from its position to the position of the conspecific (Fig. S7.2).

After the vector is established, the vector strength is initially calculated based on the distance to the conspecific (with the highest value of one found at the *attraction-range* distance and the minimum value of zero at *avoidance-range*). This is calculated as:

$$attract\text{–}strength = \frac{dist\text{–}consp - avoidance\text{–}range}{attraction\text{–}range - avoidance\text{–}range}$$

This initial vector strength value is then converted to an exponential scale using the calibrated modifier values, with the value used based on the current foraging activity of the conspecific, i.e., if the conspecific is not hunting then *attract-mod-search* is used, if it is hunting then *attract-mod-hunt* is used.

$$attract\text{–}strength = 1 - (1 - attract\text{–}strength)^{attract\text{–}mod\text{–}search}$$

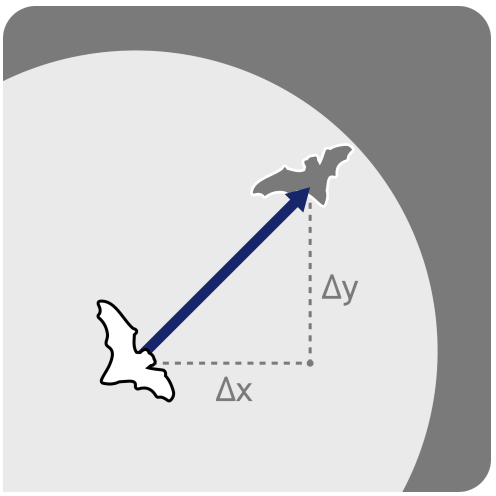


Figure S7.2. *Schematic illustrating the attraction vector determination process. The focal bat is shown in white and the attraction range visualized using a light grey circle. The conspecific is shown in grey and the drawn attraction vector in dark purple.*

7.4.2. Alignment

Bats attempt to align their movement direction with that of their conspecific while trying to maintain distance with their conspecific when drawing their alignment vector. The process of calculating the alignment vector is outlined in Figure S7.4. Aligning bats first pull the heading (*heading-consp*) and speed (*speed-consp*) of their conspecific. This is used to then predict the position where the conspecific will be after it has moved (Fig. S7.3a). As bats do not adjust their headings for the current time step until the *bats-move* submodel, which is executed later as a separate step, this location does not necessarily represent exactly where the conspecific will be after moving, but instead depicts the point where they would be if they maintained their movement direction from the previous time step.

Bats assess the current distance to their conspecific (*curr-dist*) and the distance between their current position and the predicted future position of the conspecific (*fut-dist*). The alignment vector can only point to one of two points, the two intersection points of a circle centered on the future position of the conspecific with a radius of *curr-dist* (Fig. S7.3c) and a circle centered on the focal bat with a radius of its step length (Fig. S7.3d). To locate these points, two triangles are drawn between the position of the focal bat, the future location of the conspecific, and the intersection points (Fig. S7.3e). Two lines are needed to assess the location of the intersection points, denoted as *a* and *h* in Fig. S7.3e. To estimate *a* (or *fut-dist-a*) the equation is used:

$$fut\text{–}dist\text{–}a = \frac{step\text{–}length^{2} - curr\text{–}dist^{2} + fut\text{–}dist^{2}}{2 \times fut\text{–}dist}$$

Using this value *h* (or *height*) is calculated as:

$$height = \sqrt{step\text{–}length^{2} - fut\text{–}dist\text{–}a^{2}}$$

The intersection point between these two lines (*x-mid*, *y-mid*) is then found using the current x and y position of the focal bat (*xcor* and *ycor*) as:

$$x\text{–}mid = xcor + fut\text{–}dist\text{–}a \times (fut\text{–}x\text{–}consp - xcor) \div fut\text{–}dist$$

$$y\text{–}mid = ycor + fut\text{–}dist\text{–}a \times (fut\text{–}y\text{–}consp - ycor) \div fut\text{–}dist$$

Two new x and y positions are then identified as:

$$x\text{–}new = x\text{–}mid \pm height \times (fut\text{–}y\text{–}consp - ycor) \div fut\text{–}dist$$

$$y\text{–}new = y\text{–}mid \pm height \times (fut\text{–}x\text{–}consp - xcor) \div fut\text{–}dist$$

Finally, the heading towards each of the two points from the focal animal's current position is calculated and the difference between these two headings is obtained. For animals which have a searching conspecific, the point is taken which results in a heading which has a smaller deviation from the conspecific's current heading (*heading-consp*). If the conspecific is hunting, a check first occurs to see if the difference in the current headings between the two bats is greater than 90 degrees (meaning that they are currently heading in nearly opposing directions). If so, bats then take the point which is closer to the inverse of *heading-consp*. This allows bats to continue their path around a hunting bat, rather than mimicking the erratic turning angles that hunting bats exhibit. The selected new position is then used to draw the alignment vector (Fig. S7.3f).

Some conditions may exist where this math is not possible. One being that the circles are separate. This can occur when the *step-length* of the focal animal is too small to create a circle which intersects the circle around the future position of the conspecific. When this occurs, bats instead draw a vector straight towards the future position of the conspecific with a length of their *step-length*. This will not maintain distance but can minimize increases in distance. Additionally, circles can be concentric causing no possible solutions, or be coincident with an infinite number of solutions. In both of these cases the alignment vector is set to (0,0) and alignment does not occur.

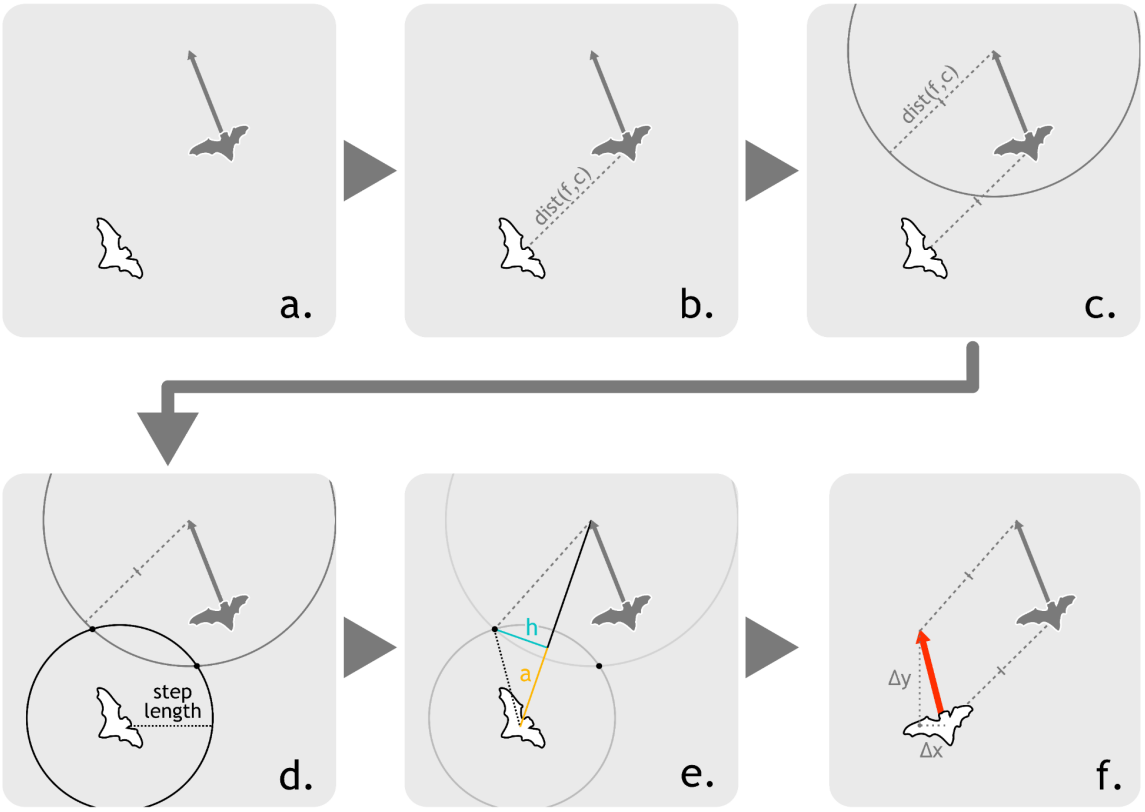


Figure S7.3. *Schematic illustrating the alignment vector determination process. The focal bat is shown in white and the conspecific is shown in grey. The circles in c. and d. are shown only for illustrative purposes and are not explicitly calculated. In panel e. the blue and yellow lines represent height and fut-dist-a described in the main text. In f. the resulting alignment vector is shown in orange.*

After the alignment vector is established, the vector strength is calculated based on the distance to the conspecific. Alignment differs from the other two processes in that it is strongest (a value of one) at the *alignment-range* and then decreases to zero in both directions, so that it has a value of zero at both the *attraction*- and *avoidance-ranges* (Fig. S7.4). The initial *align-strength* value is calculated as:

$$align\text{–}strength \; =$$

$$\frac{attraction\text{–}range - dist\text{–}consp}{attraction\text{–}range - alignment\text{–}range}, \qquad if \; alignment\text{–}range < dist\text{–}consp < attraction\text{–}range$$

$$\frac{dist\text{–}consp - avoidance\text{–}range}{alignment\text{–}range - avoidance\text{–}range}, \qquad if \; avoidance\text{–}range < dist\text{–}consp < alignment\text{–}range$$

This weight is then converted to an exponential scale as in the *Attraction* submodel using the calibrated modifier values, *align-mod-search* for bats with searching conspecifics and *align-mod-hunt* for bats with hunting conspecifics, as:

$$align\text{–}strength = 1 - (1 - align\text{–}strength)^{align\text{–}mod\text{–}search}$$

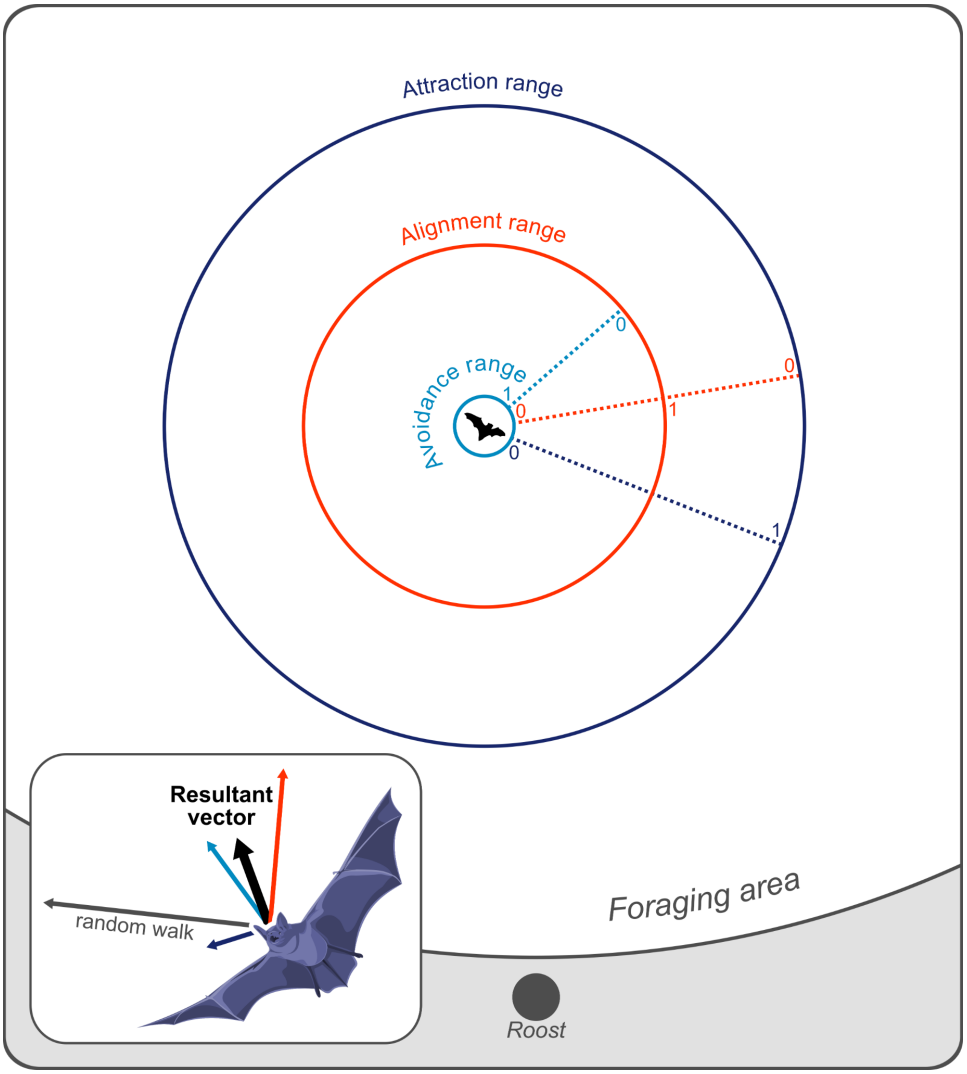


Figure S7.4. *The ranges of each of the three interaction-based movement processes and the location of their minimum (0) and maximum (1) strength value. Each vector is independently assessed and the resulting vector is taken as a mixture of all four movement vectors (including random walk) (inset figure).*

7.4.3. Avoidance

Avoidance is used by bats to avoid collisions and potential echolocation jamming (Amichai et al., 2015) from neighboring bats. The avoidance vector is essentially

calculated as the inverse of attraction. Bats first pull the x and y positions of their conspecific to get a vector heading towards the bat. Then this vector is flipped to head directly away from the conspecific (Fig. S7.5).

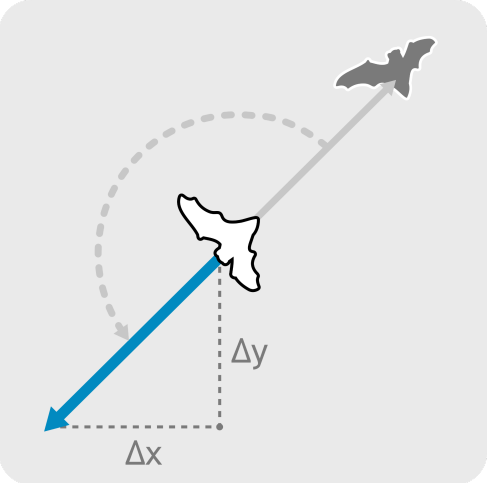


Figure S7.5. *Schematic illustrating the avoidance vector determination process. The focal bat is shown in white and the conspecific is shown in grey. The blue arrow represents the avoidance vector.*

The initial strength of the avoidance vector is then calculated using the distance from the conspecific where the highest value (1) occurs at the *avoidance-range* and then decreases to zero at the *alignment-range*. Avoidance is set to zero at ranges greater than the *alignment-range*. This strength is again modified exponentially, as for the attraction and alignment vectors, using the calibrated parameters *avoid-mod-search* (when conspecific is searching) and *avoid-mod-hunt* (when the conspecific is hunting), using the equation:

$$avoid\text{–}strength = 1 - (1 - avoid\text{–}strength)^{avoid\text{–}mod\text{–}search}$$

An exponential scale was used for the vector strengths to calibrate the movement processes to the data in a way in which the overall magnitudes of each process was the same, but the distances at which the maximum value of one occurred changed between processes (Fig. S7.6). This approach modified only the shape of the curve rather than the range of magnitudes of the vector strengths.

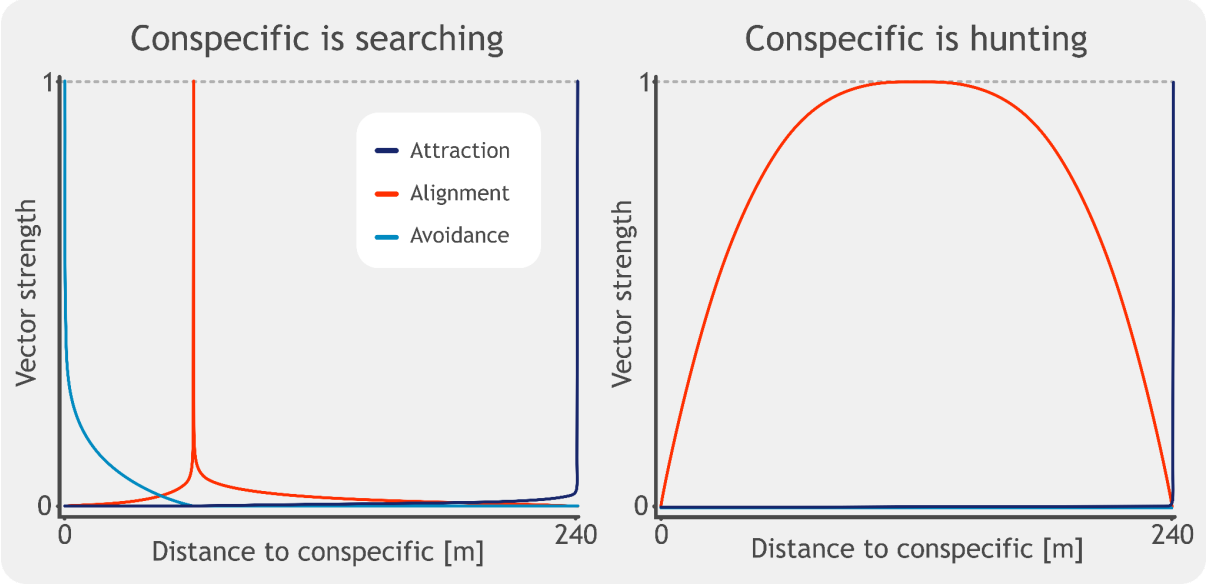


Figure S7.6. *The relative strengths of each of the three interaction-based movement processes depending on the distance from the focal bat to its conspecific (in m) and whether the conspecific is or is not hunting. The shape of the vector strength relationship is determined by the calibrated parameters described in the Attraction, Alignment, and Avoidance sections.*

7.4.4. Random walk (random-walk)

The random walk submodel is the only movement process calculated for all searching bats in every time step. This process entirely controls the movement direction of bats when they have no conspecific or when running the null model. Both correlated and biased random walk behavior is used in the process. Bats select either the center of the foraging area or their previous heading to target when estimating their movement direction, based on their current distance to the center of the foraging area. When well within the foraging area (< *foraging-radius* - 20 cells) bats use their previous heading to target their random walk behavior, while when completely outside of the foraging area (> *foraging-radius* + five cells), bats target a point that is 10% determined by targeting the center of the foraging area and 90% by their previous heading. For bats at a distance in the middle of these two ranges, the targeting is blended between the two directions (Fig. S7.7). The ranges for targeting behavior were selected as they allowed animals to thoroughly cover the foraging area when moving in the landscape without often running into the landscape walls. After the random walk target is determined, a random turning angle is then added to add stochasticity to the movement direction (see ODD Element 4.9).

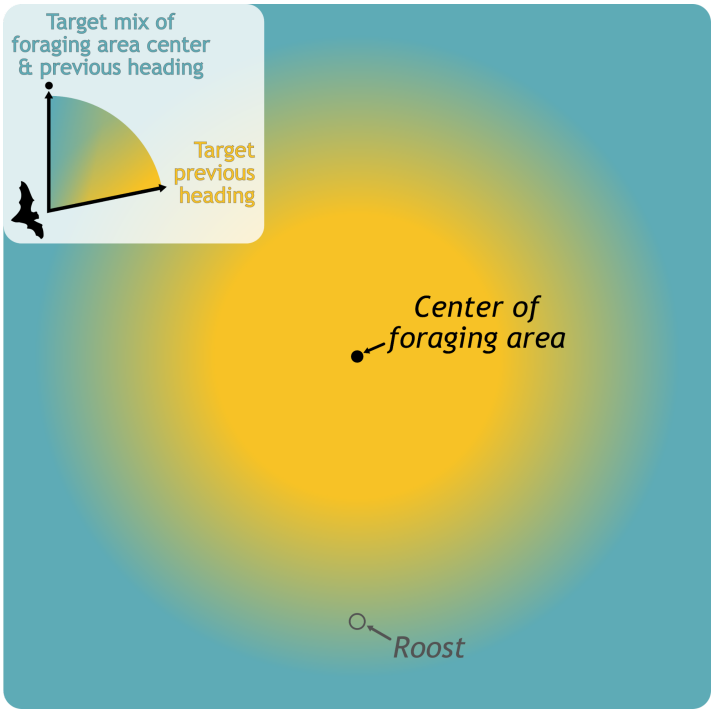


Figure S7.7. *In the random walk submodel, bats select a heading target based on their current distance to the center of the foraging area. When outside of the foraging area, this target is set to a mix of the foraging area center (10%) and the previous heading of the bat (90%) (region shown in blue), while when well within the foraging area bats use only their previous heading as their target (yellow). In between these areas the targeting is blended between the two directions (gradient region). Additional wiggle is added after this targeting step before the random walk vector is calculated.*

Bats calculating their random walk vector begin by first selecting their two targets (*heading-rw-head* for previous heading and *heading-rw-forg* for the center of the foraging area). Then the relative weight of the two targets is calculated linearly between the inner and outer range values based on the bats current distance to the center of the foraging area. The heading is then determined using the direction to the two targets and the calculated relative weight. The random contribution to the turning angle is then pulled from a gamma distribution fit to the turning angles of searching bats in the empirical dataset. To avoid the additional contribution from interactions between bats causing the realized turning angle of bats to deviate from observed turning angles, the contribution of additional stochasticity in turning angles is reduced by 55% for bats which currently have a conspecific. This value was selected as it resulted in total turning angles (determined as the change in heading between consecutive time steps) which resembled those of the empirical bats (Fig. S7.8). Bats then add the random component to the heading based on targeting and draw their

random walk vector using this combined heading and their step length value. The strength of this vector is then set as a flat value using the calibrated parameters *rw-mod-search* (when conspecific is searching) and *rw-mod-hunt* (when the conspecific is hunting).

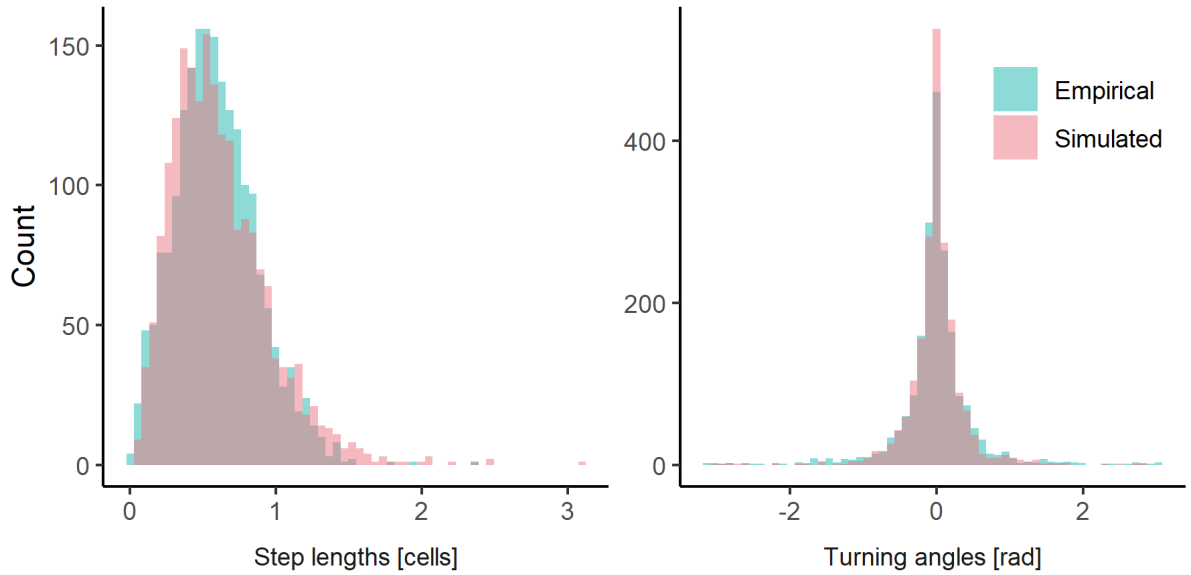


Figure S7.8. *Comparison of empirical (blue) and simulated (red) step lengths and total turning angles of searching bats. For both empirical and simulated bats 2000 data points are shown.*

7.5. Update heading and fly (bats-move)

In this submodel, bats calculate their resulting movement direction based on the four movement vectors and their strengths, and then move. To do this, bats calculate the changing x and y components of their resulting movement direction separately. To calculate the x component, bats multiply the x value from each of the vectors by its corresponding vector strength and then add together each of the four resulting values. The same is done for the y values and then the heading is updated using the resulting vector. The calculated heading is compared to the previous heading to determine the turning angle bats take in a time step (*turn-angle-real*).

Bats which have triggered the *flying-towards-food* behavior (meaning that they have encountered an occupied food cell with connected unoccupied food cells; details in the *food-check* procedure), skip the above calculations and instead face the location of their targeted cell.

All bats which have not found food then fly in the direction of their updated heading at their flying speed (i.e., *step-length*) and check for food (*food-check* submodel). This

step is broken into three parts to reduce the chance that a bat would fly directly through and miss a food cell, so bats take a step equal to their *step-length* divided by three and check for food, then repeat this process two more times.

7.6. Hunting bat flight (hunting-fly)

Bats which have located a food cell fly around the cell to simulate hunting behavior. This is executed using the same mechanics as the ***random-walk*** submodel, only the center of the food cell is used rather than the center of the foraging area as a target and the ranges are changed. In this submodel, the target is entirely set to the center of the food cell when the bat is at a distance greater than 0.5 cells away from the center of the food cell, while when less than 0.1 cells away from the center bats use only their previous heading as their target. The gamma distributions used to pull the turning angle and step length values used here were fit to data from hunting bats in the empirical dataset.

7.7. Check if food has been found (food-check)

In this submodel bats check their surroundings to locate a food cell. This process starts by first creating a temporary variable called *food-found-this-tick* and setting it to “false“. This variable will be used later to update the state variables of bats which have found food. Bats then check if the cell that they are currently in has food which has not been found by another bat (i.e., *found* = “false“). If so, then bats move to the center of that patch and flip the *food-found-this-tick* boolean to “true“.

If there is food in the cell, but it is occupied by another bat, bats then use the *patch-id* to identify if there is another cell in the patch which has food and is unoccupied. This was done as bats are capable of detecting prey once they are in a food patch, so simulated bats were allowed to move within the swarms to locate a free cell within the patch, if available. If so, bats target one of these free cells (*targ-cell*) and set *flying-towards-food* boolean to “true”.

If the cell instead does not contain food, bats check for food in a 15m radius around their position. If an unoccupied food cell is found, then bats target this food cell and set *flying-towards-food* boolean to “true”. If more than one unoccupied food cell is found, then the target cell is selected randomly. If an occupied food cell is found in the radius, then bats go through the above process for checking for unoccupied food cells in the patch containing the occupied food cell.

When bats have found food they update a suite of variables: *food-found* is set to “true”, *conspecific* is set to “nobody”, they record the location of the food cell they

now occupy (*food-cell*), set their color to white, ask the cell to set its *found* value to "true", set *flying-towards-food* to "false", and add the number of time steps that it took them to find food (adjusted for the time they left the roost) to the outputs list *food-found-ticks-list*.

If a bat finds food, it alerts any bats which see it as their conspecific at that time step. If there are unoccupied food cells in the patch it found, then the other bat will target an unoccupied food cell in the patch and set their *flying-towards-food* boolean to "true".

For bats which reach their *targ-cell* but find it occupied, they reset their *flying-towards-food* boolean to "false".

7.8. Calibration and evaluation

Calibration**:** To ensure that the behavior of and interactions between modelled bats reflected the movement patterns found for the real bats in the study, we calibrated parameters controlling the vector strength for each of the four movement processes and the alignment range distance using an inverse modelling approach (Kramer-Schadt et al., 2007; Railsback & Grimm, 2019). The calibration was broken up into two steps with the movements of bats with and without hunting conspecifics being calibrated separately. Two related patterns were used for each step: 1) the relationship between initial distance and changes in distance between focal bats and their nearest conspecific (main text Fig. 4c & d), and 2) the overall shape of the density curve fit to the distance changes. A total of 10 parameters were calibrated, with the calibrated parameters for bats with searching conspecifics being: *attract-mod-search*, *align-mod-search*, *avoid-mod-search*, *rw-mod-search*, and *alignment-range-search*; and for bats with hunting conspecifics were: *attract-mod-hunt*, *align-mod-hunt*, *avoid-mod-hunt*, *rw-mod-hunt*, and *alignment-range-hunt*.

To calibrate these parameters, for each state (conspecific searching vs. hunting) 25 simulations were run using each possible parameter combination. The calibration for bats with searching conspecifics was run first and then the second calibration was run for bats with hunting conspecifics. In each simulation, the number of landscape patches (*no-patches*) was randomly selected, ranging from spatial aggregation levels of 0 - 100%. This was done as the total spatial aggregation of food resources in the real environment was unknown. For both calibrations 80 bats were generated and a number of these bats were tracked depending on the state, with 20 bats tracked for the first calibration and 80 tracked for the second. The number differed as a great

deal more points were collected in the first calibration, due to the greater amount of time spent by bats networking with searching bats.

Tracked bats executed the *movement-outputs* procedure to record their movement behavior once per every four time steps (32 seconds). These outputs were only collected when bats met certain criteria: they must not have found food yet (*food-found* = "false"), they must not currently be flying towards an unoccupied food cell (*flying-towards-food* = "false"), and they must have already left the roost (*leave-roost-tick* < time step). If these conditions are met, bats then calculate their change in distance (*diff-dist*) as the difference between their current distance to the conspecific they had in the previous recording (called *prev-consp*) and the distance they were to that conspecific in the previous recording (*prev-dist-c*). This method was taken to collect outputs in the same manner as they were calculated for bats in the empirical dataset. Bats then record *prev-dist-c* and *diff-dist* in an output list. After recording, bats check if they have a conspecific and, if so, save the identity (*prev-consp*), foraging behavior (*prev-consp-hunt*), and distance to the conspecific (*prev-dist-c*) at the end of the procedure.

Each calibrated parameter was varied over four levels and all combinations of these four levels were tested. Ranges were identified through exploratory runs which narrowed the values tested (see Table S7.8.1). The calibration was run using the *BehaviorSpace* feature in NetLogo v6.2.0 (Shargel & Wilensky, 2002).

Table. S7.8.1. *Values tested for calibrated parameters.*

Parameter	***Code***	**Minimum**	**Maximum**	**Interval**
Attraction modifier - Searching	*attract-mod-search*	0.001	0.016	0.005
Attraction modifier - Hunting	*attract-mod-hunt*	0	0.0015	0.0005
Alignment modifier - Searching	*align-mod-search*	0.01	0.04	0.01
Alignment modifier - Hunting	*align-mod-hunt*	1	7	2
Avoidance modifier - Searching	*avoid-mod-search*	0.05	0.2	0.05
Avoidance modifier - Hunting	*avoid-mod-hunt*	0	0.15	0.05
Random walk modifier - Searching	*rw-mod-search*	0.3	0.6	0.1
Random walk modifier - Hunting	*rw-mod-hunt*	0	0.15	0.05
Alignment range - Searching	*alignment-range-search*	30	120	30
Alignment range - Hunting	*alignment-range-hunt*	30	120	30

The outputs were processed in R statistical software v4.0.3 (R Core Team, 2021) and pooled based on their parameter combination. The first pattern was assessed in the same manner as the empirical patterns by fitting a 3rd order polynomial model to 5000 points from each parameter combination using the *lmer* function in the "lme4" package (Bates et al., 2021, p. 4). Using the *ggeffect* function in the "ggeffects" package (Lüdecke et al., 2021), the resulting statistical model predictions were converted into a table with a row for each initial distance value between 0 and 240m at 10m increments. The deviation between the empirical and simulated statistical models was then estimated at each initial distance value and the root mean squared error (RMSE) was calculated for each parameter combination.

For the second pattern, density values were calculated using the built-in *density* function in R at 100 equally spaced points between 0 and 350m for both the empirical and simulated differences in distances recorded. Again 5000 points were used per parameter combination. At each point, the deviation between the empirical and simulated density curves was determined and the overall error was again calculated as RMSE.

The relative fit of the results from each parameter combination to each pattern was calculated as a ranking which increased with increasing RMSE, e.g., the parameter combination resulting in the tightest fit (i.e., lowest RMSE) held a rank of 1. These two ranks were added together and the parameter combination which yielded the lowest overall rank was selected. Calibration resulted in successful fits of the movement processes to the empirical patterns, with average model predictions for the first pattern falling within the 95% confidence intervals for both states at all distances. The selected parameter values can be found in Table S7.1 and resulting fits in Figure S7.9 below.

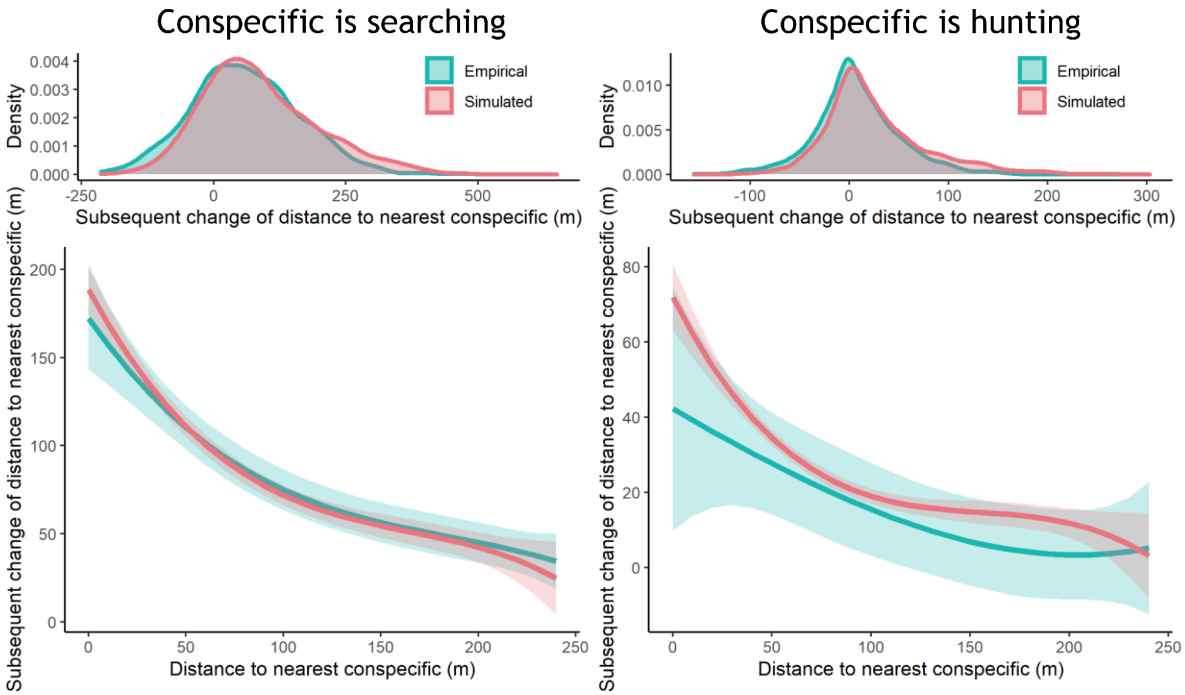


Figure S7.9. *Calibration results for the calibration movement process, showing comparison between empirical (blue) and simulated (red) patterns of movement behavior for the best fitting parameter values. The patterns in the relationship between initial distance and changes in distance between focal bats and their nearest conspecific (bottom panels) and overall shape of the density curves fit to the distance changes (top panels) are shown for both bats with searching (left) and hunting (right) conspecifics. Shaded regions in the bottom panel denote 95% confidence intervals.*

The null model was additionally run (*null-model?* set to “true”) using the same simulation specifications to assess the pattern outputs for bats which are not networking (Fig. S7.10).

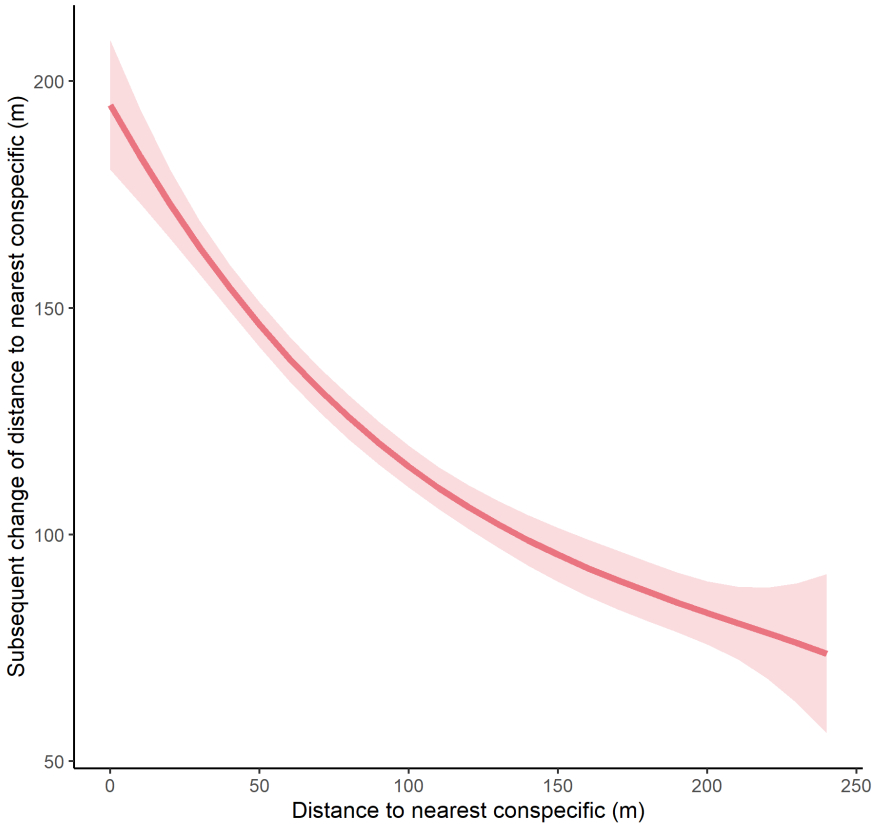


Figure S7.10. *The relationship between initial distance and changes in distance between focal bats and their nearest neighbor for the null model. The shaded region denotes the 95% confidence interval.*

Evaluation: To evaluate the calibrated model's ability to emulate bat movement features, we compared model outputs to three population-level bat movement patterns. The three patterns we used were distributions of the: 1) euclidean distance between the bat starting point and its final point at a food cell (beeline), 2) beeline divided by the sum of euclidean distances between each 8 sec time step (straightness index), and 3) time difference between the bat leaving the roost and finding a food cell (time to first forage). It is important to note that these comparisons are independent, or secondary predictions, and did not include any additional parameterization or re-calibration.

To compare these patterns, the x and y positions of simulated bats were collected once per time step for bats in landscapes with varying spatial aggregation levels (1, 2, 4, 8, 16, 32, 64, 128, or 213 patches). While 80 bats were simulated in the landscape (approximately the empirical colony size), only four bats were tracked in each simulation. Twenty-five simulations were run per spatial aggregation level, totalling in 100 tracked bats per level.

The model outputs and empirical data were then analyzed following the same methodology using the "sp" package (Pebesma et al., 2021) in R statistical software v4.0.2 (R Core Team, 2021). The patterns were then visually compared to evaluate fit at each spatial aggregation level tested (Fig. S7.11).

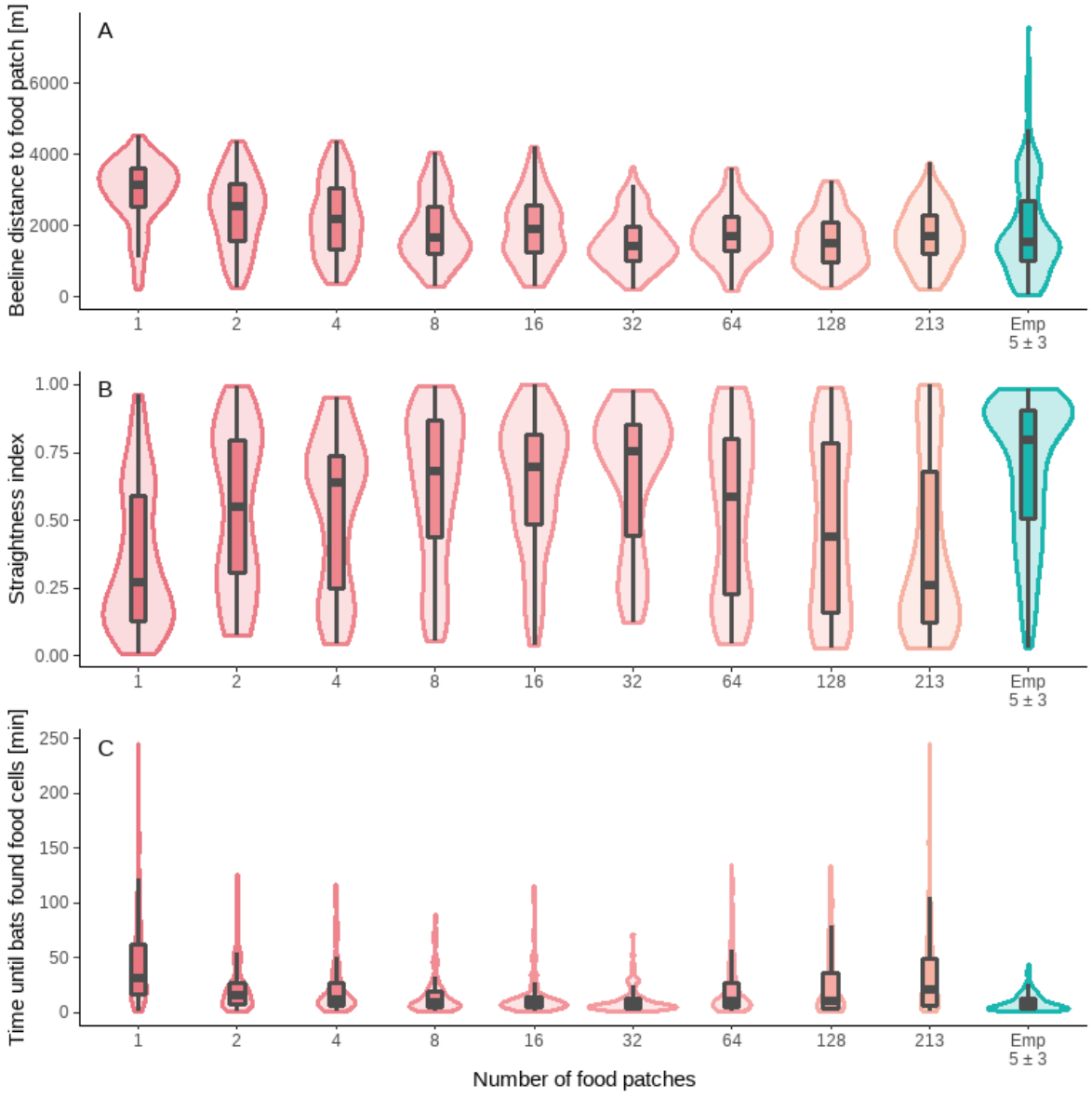


Figure S7.11. Evaluation outputs for the three patterns tested, beeline (A), straightness index (B), and time to first forage (c). Model outputs are shown in red tones for all tested spatial aggregation levels and the corresponding empirical patterns are shown in blue. Distributions of outputs are visualized using filled regions and the boxplot horizontal line denotes the median while upper and lower hinges represent the 25th and 75th percentiles, respectively.

The fit of the evaluation outputs varied depending on the tested spatial aggregation level. For the beeline pattern, most median values for the model outputs fell within the interquartile range of the empirical data. The only exception occurred when only a single food patch was generated, where the beeline value was recorded as being higher than the empirical pattern. The empirical observations showed a much wider spread of values, reaching distances substantially higher than recorded in the model (7584m), potentially due to natural variations in the location of food patches in the real landscape.

Straightness index produced a U-shaped trend, with higher straightness values found at intermediate patch numbers and lower values exhibited at both low and high patch numbers. For this pattern, landscapes with between 2 and 64 patches resulted in median values which fell within the interquartile range of the empirical pattern. Though when assessing the overall shape of the distribution, landscapes with between 8 and 32 patches were observed to more similarly fit the shape of the empirical data, with a higher density of points occurring towards a straightness index value of 1.0.

The model also resulted in a U-shaped trend for the time to first forage pattern, with both a higher median value and higher variability in output values found for low and high numbers of patches when compared to outputs for intermediate patch levels. The empirical observations predominantly occurred at very short time values, with the mean falling at only 8.8 minutes. The median values for landscapes containing between 4 and 128 patches all fell within the interquartile range of the empirical observations, with 32 patches most similarly fitting the overall distribution of the empirical pattern.

Overall the model was capable of producing outputs similar to real bat observations from the tracking study, with the best fit occurring at intermediate spatial aggregation levels for all three tested patterns.

References

Amichai, E., Blumrosen, G., & Yovel, Y. (2015). Calling louder and longer: How bats use biosonar under severe acoustic interference from other bats. *Proceedings of the Royal Society B: Biological Sciences*, *282*(1821), 20152064. https://doi.org/10.1098/rspb.2015.2064

Bates, D., Maechler, M., Bolker, B., Walker, S., Christensen, R. H. B., Singmann, H., Dai, B., Scheipl, F., Grothendieck, G., Green, P., Fox, J., Bauer, A., & Krivitsky, P. N. (2021). *lme4: Linear Mixed-Effects Models using "Eigen" and S4* (1.1-27.1)

[Computer software]. https://CRAN.R-project.org/package=lme4
Boonman, A., Fenton, B., & Yovel, Y. (2019). The benefits of insect-swarm hunting to echolocating bats, and its influence on the evolution of bat echolocation signals. *PLOS Computational Biology*, *15*(12), e1006873. https://doi.org/10.1371/journal.pcbi.1006873
Cvikel, N., Egert Berg, K., Levin, E., Hurme, E., Borissov, I., Boonman, A., Amichai, E., & Yovel, Y. (2015). Bats Aggregate to Improve Prey Search but Might Be Impaired when Their Density Becomes Too High. *Current Biology*, *25*(2), 206–211. https://doi.org/10.1016/j.cub.2014.11.010
Gager, Y. (2019). Information transfer about food as a reason for sociality in bats. *Mammal Review*, *49*(2), 113–120. https://doi.org/10.1111/mam.12146
Kalko, E. K. V. (1995). Insect pursuit, prey capture and echolocation in pipestirelle bats (*Microchiroptera*). *Animal Behaviour*, *50*(4), 861–880. https://doi.org/10.1016/0003-3472(95)80090-5
Kramer-Schadt, S., Revilla, E., Wiegand, T., & Grimm, V. (2007). Patterns for parameters in simulation models. *Ecological Modelling*, *204*(3), 553–556. https://doi.org/10.1016/j.ecolmodel.2007.01.018
Lüdecke, D., Aust, F., Crawley, S., & Ben-Shachar, M. S. (2021). *ggeffects: Create Tidy Data Frames of Marginal Effects for "ggplot" from Model Outputs* (1.1.1) [Computer software]. https://CRAN.R-project.org/package=ggeffects
Pebesma, E., Bivand, R., Rowlingson, B., Gomez-Rubio, V., Hijmans, R., Sumner, M., MacQueen, D., Lemon, J., Lindgren, F., O'Brien, J., & O'Rourke, J. (2021). *sp: Classes and Methods for Spatial Data* (1.4-5) [Computer software]. https://CRAN.R-project.org/package=sp
R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/
Railsback, S. F., & Grimm, V. (2019). *Agent-based and individual-based modeling: A practical introduction*. Princeton university press.
Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25–34. https://doi.org/10.1145/37402.37406
Shargel, B., & Wilensky, U. (2002). *BehaviorSpace.* Northwestern University.
Voigt, C. C., Russo, D., Runkel, V., & Goerlitz, H. R. (2021). Limitations of acoustic monitoring at wind turbines to evaluate fatality risk of bats. *Mammal Review*. https://doi.org/10.1111/mam.12248