Biomimicry of Bacterial Foraging
for Distributed Optimization and Control

Kevin M. Passino[1]
Presented by: Alexander Van de Kleut[2]

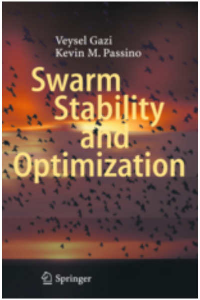
[1]The Ohio State University
Electrical and Computer Engineering

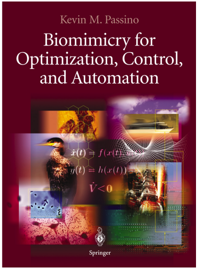
[2]University of Waterloo
Centre for Theoretical Neuroscience

IEEE Control Systems Magazine, 2002

Table of Contents

About the Author





About the Author

Fuzzy control KM Passino, S Yurkovich, M Reinfrank Addison-wesley 42, 15-21, 1998	3599
Biomimicry of bacterial foraging for distributed optimization and control KM Passino IEEE control systems magazine 22 (3), 52-67, 2002	3023
Stability analysis of swarms V Gazi, KM Passino IEEE transactions on automatic control 48 (4), 692-697, 2003	1125
Stable adaptive control using fuzzy systems and neural networks JT Spooner, KM Passino IEEE Transactions on Fuzzy Systems 4 (3), 339-359, 1996	728
Stability analysis of social foraging swarms V Gazi, KM Passino IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 …, 2004	710

Foraging

Foraging

- searching for nutrients
- avoiding noxious stimuli (toxins, predators, etc)

Social Foraging

- increases likelihood of finding nutrients
- better detection and protection from noxious stimuli
- gains can offset cost of food competition

How can we view foraging as an Optimization Process?

- We have some parameters θ and a loss function $J(\theta)$ that we want to minimize

Foraging as Optimization

How can we view foraging as an Optimization Process?

- We have some parameters θ and a loss function $J(\theta)$ that we want to minimize
- θ can represent the position of an organism in its environment

Foraging as Optimization

How can we view foraging as an Optimization Process?

- We have some parameters θ and a loss function $J(\theta)$ that we want to minimize
- θ can represent the position of an organism in its environment
- J can represent the concentration of nutrients and noxious stimuli
  - smaller values of J = more nutrients, less noxious stimuli
  - higher values of J = more noxious stimuli, less nutrients

Foraging as Optimization

How can we view foraging as an Optimization Process?

- We have some parameters θ and a loss function $J(\theta)$ that we want to minimize
- θ can represent the position of an organism in its environment
- J can represent the concentration of nutrients and noxious stimuli
  - smaller values of J = more nutrients, less noxious stimuli
  - higher values of J = more noxious stimuli, less nutrients
- In general, J and θ can be arbitrary
  - $\theta \in \mathbb{R}^p$
  - $J : \mathbb{R}^p \to \mathbb{R}$

E. coli

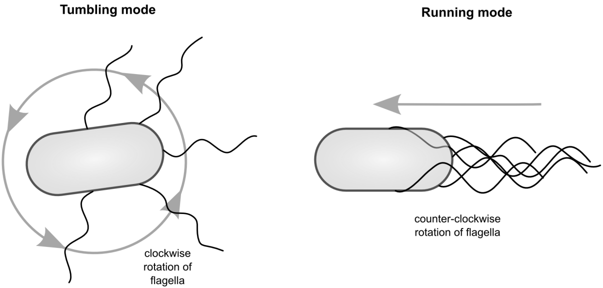
- Model organism
  - Highly studied
  - Well-characterized foraging behaviour
  - Probably won't feel bad about simplifying its behaviour

E. coli

- Model organism
  - Highly studied
  - Well-characterized foraging behaviour
  - Probably won't feel bad about simplifying its behaviour
- Social organism
  - Secretes signals to attract others nearby
  - Encourages "swarming" or "clumping"

E. coli Behaviour

- Swims using left-handed helical flagella ("propellers")
  - **Tumble**: flagella all rotate clockwise → pull on cell in all directions → random movement
  - **Run**: flagella all rotate counterclockwise → flagella form a bundle → push on cell in one direction → directed movement



E. coli Behaviour

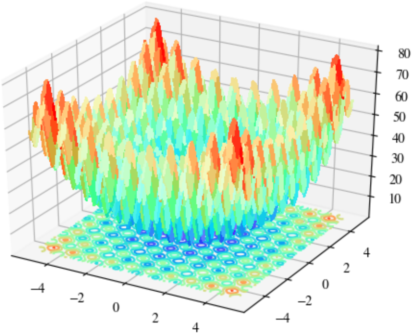
- If during a tumble *E. coli* swims down a nutrient concentration gradient:
  - Prolongs time spent on a run
  - Continues moving in the same direction
- Otherwise:
  - Tends to switch to a tumble (search for more)
  - Moves randomly which searching for more nutrient gradients to exploit
- Call a tumble followed by a run a “chemotaxis step”

Algorithm for a Single Bacterium

for $j \leftarrow 1 \dots N_c$ **do**:
    $\phi \sim S^p$
    $\theta \leftarrow \theta + c\phi$
    while $J(\theta + c\phi) < J(\theta)$ **do**:
        $\theta \leftarrow \theta + c\phi$

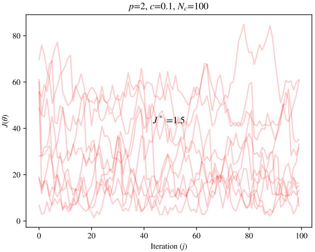
- θ: p-dimensional vector (randomly initialized)
- N_c: number of chemotaxis steps
- $\phi \sim S^p$: a random p-dimensional unit vector
- c: a step-size

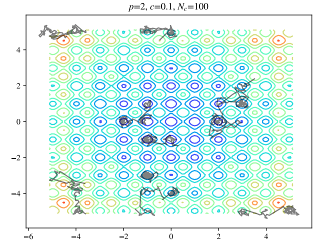
Loss Function to Optimize



$$J(\theta) = An + \sum_{i=1}^{n} \left(x_i^2 - A\cos(2\pi x_i) \right)$$

Results of Single Bacterium





- Relatively inconsistent performance for a highly nonconvex function.

Algorithm for a Colony

for $j \leftarrow 1 \ldots N_c$ **do**:
  for $i \leftarrow 1 \ldots S$ **do**:
    $\phi \sim S^p$
    $\theta_i \leftarrow \theta_i + c_i\phi$
    while $J(\theta_i + c_i\phi) + J_{cc}(\theta_i + c_i\phi) < J(\theta_i) + J_{cc}(\theta_i)$ **do**:
      $\theta_i \leftarrow \theta_i + c_i\phi$

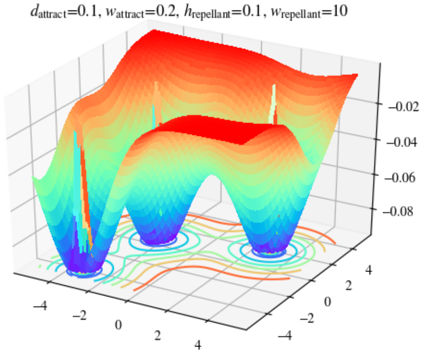
- θ_i: ith p-dimensional vector (randomly initialized)
- S: number of bacteria in the colony
- c_i: a step-size for bacterium i
- $J_c c$: cell-to-cell interactions

J_{cc} and swarming behaviour

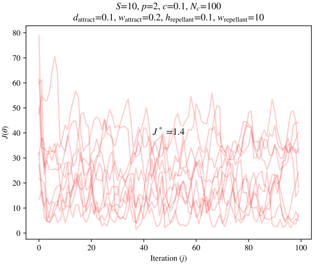
- *E. coli* do social foraging
- Secrete a substance to indicate to attract nearby *E. coli* and encourage swarming and biofilm formation
- Strength of signal diffuses over space
- Also want to avoid crowding
- Use sum of two Gaussian functions to model this

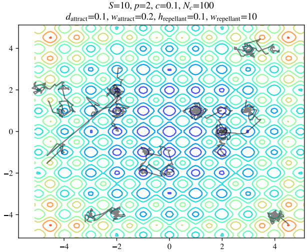
$$J_{cc}(\theta) = \sum_{i=1}^{S} - d_{\text{attract}} \exp\left(-w_{\text{attract}}(\theta - \theta_i)^T(\theta - \theta_i)\right) \\ + h_{\text{repellant}} \exp\left(-w_{\text{repellant}}(\theta - \theta_i)^T(\theta - \theta_i)\right)$$

J_{cc} and swarming behaviour



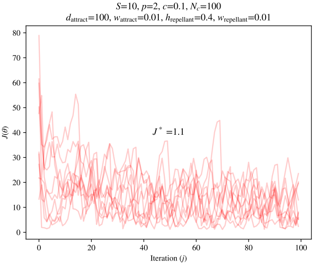
Results of Colony with Swarming

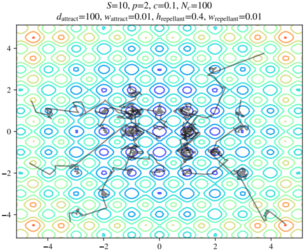




- Still relatively inconsistent performance for a highly nonconvex function.
- But wait... What if the problem is just the hyperparameters?

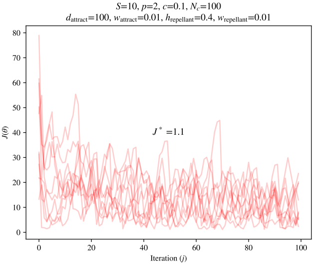
Results of Colony with Swarming

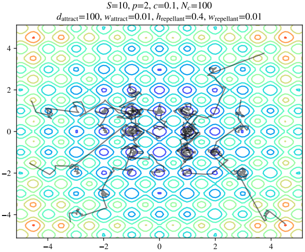




- By trying out different combinations of hyperparameters we can improve overall performance.
- Here we increased the depth and width of attraction as well as the depth and width of repellance to increase "global" behaviour.

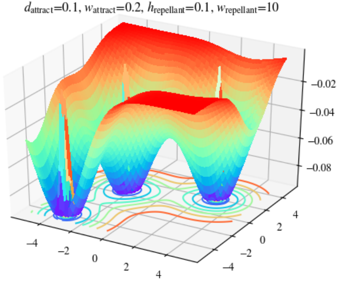
Results of Colony with Swarming

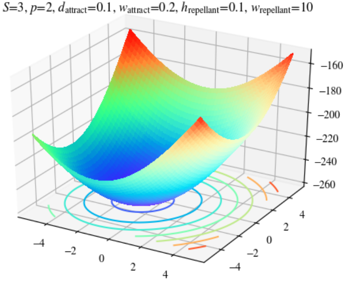




- By trying out different combinations of hyperparameters we can improve overall performance.
- Here we increased the depth and width of attraction as well as the depth and width of repellance to increase ”global” behaviour.
- Important to know scale of J relative to scale of J_{cc} for tradeoff.
    - Can think of this like hyperparameters for PSO

Comparing J_{cc}





E. coli reproduction

- *E. coli* “reproduce” via
  1. **Binary fission**: essentially creating a clone
  2. **Horizontal Translation**: merging genetic material with others

E. coli reproduction

- *E. coli* “reproduce” via
  1. **Binary fission**: essentially creating a clone
  2. **Horizontal Translation**: merging genetic material with others
- Algorithm designed to mimic binary fission
  - More fit individuals more likely to survive
  - Less fit individuals more likely to die

E. coli reproduction

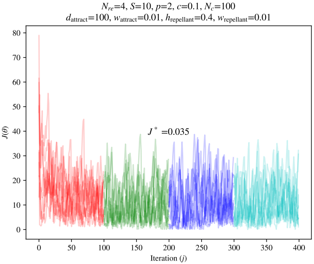
- *E. coli* "reproduce" via
  1. **Binary fission**: essentially creating a clone
  2. **Horizontal Translation**: merging genetic material with others
- Algorithm designed to mimic binary fission
  - More fit individuals more likely to survive
  - Less fit individuals more likely to die
- Horizontal translation could be incorporated (like a genetic algorithm)

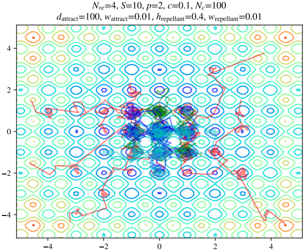
Algorithm for a Reproducing Colony

for $k \leftarrow 1 \ldots N_{re}$ **do**:
    for $j \leftarrow 1 \ldots N_c$ **do**:
        for $i \leftarrow 1 \ldots S$ **do**:
            $\phi \sim S^p$
            $\theta_i \leftarrow \theta_i + c_i\phi$
            while $J(\theta_i + c_i\phi) + J_{cc}(\theta_i + c_i\phi) < J(\theta_i) + J_{cc}(\theta_i)$ **do**:
                $\theta_i \leftarrow \theta_i + c_i\phi$
    delete worst $S/2$ and reproduce best $S/2$

- N_{re}: number of reproduction steps

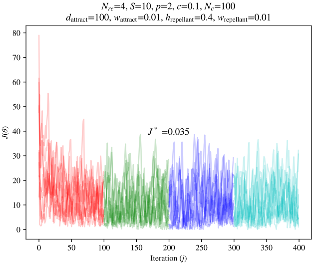
Results of Reproducing Colony

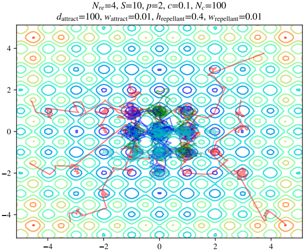




- Individuals with higher values of J killed off

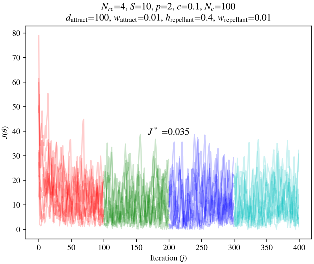
Results of Reproducing Colony

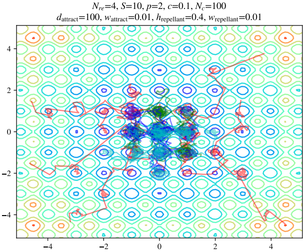




- Individuals with higher values of J killed off
- Individuals with lower values of J duplicated
  - Ideally move away due to repellance

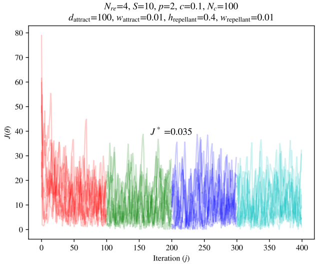
Results of Reproducing Colony

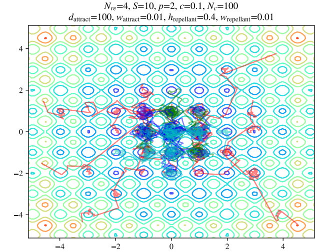




- Individuals with higher values of J killed off
- Individuals with lower values of J duplicated
  - Ideally move away due to repellance
- Idea is to encourage searching in space nearby "best" individuals

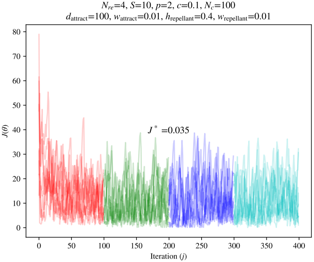
Results of Reproducing Colony

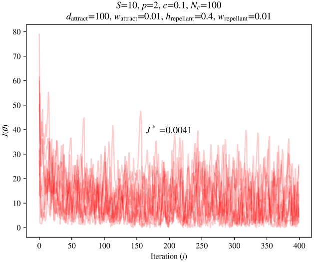


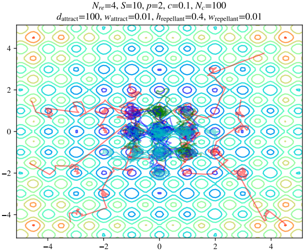


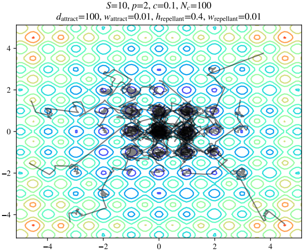
- Individuals with higher values of J killed off
- Individuals with lower values of J duplicated
  - Ideally move away due to repellance
- Idea is to encourage searching in space nearby “best” individuals
- If repellance isn’t high enough then repeated iterations of evolution can concentrate colony in local minimum

Does Reproduction Help?

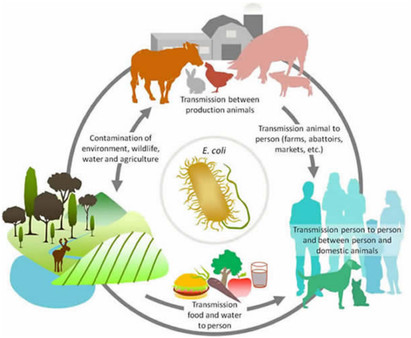






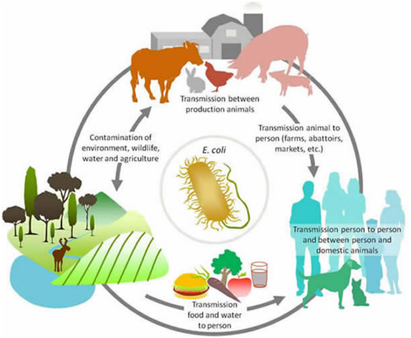


Elimination-Dispersal Events



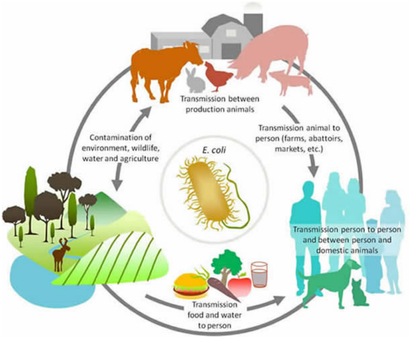
- Over time, random events disperse populations of *E. coli*
  - Water, animal activity, human intervention

Elimination-Dispersal Events



- Over time, random events disperse populations of *E. coli*
  - Water, animal activity, human intervention
- May destroy chemotactic progress
  - But may also bring *E. coli* to good food sources

Elimination-Dispersal Events



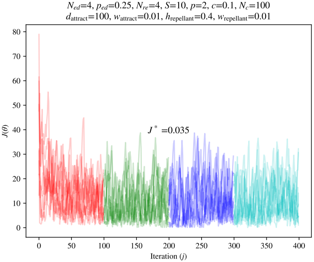
- Over time, random events disperse populations of *E. coli*
  - Water, animal activity, human intervention
- May destroy chemotactic progress
  - But may also bring *E. coli* to good food sources
- For optimization, this is a method to prevent stagnation and move out from local minima

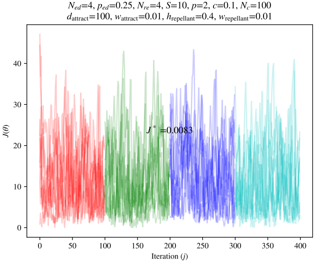
Algorithm for a Dispersing Colony

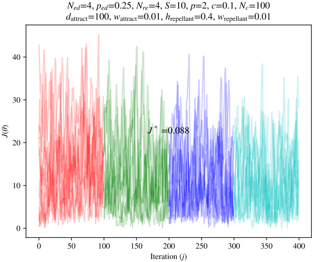
for $l \leftarrow 1 \ldots N_{ed}$ **do**:
    for $k \leftarrow 1 \ldots N_{re}$ **do**:
        for $j \leftarrow 1 \ldots N_c$ **do**:
            for $i \leftarrow 1 \ldots S$ **do**:
                $\phi \sim S^p$
                $\theta_i \leftarrow \theta_i + c_i\phi$
                while $J(\theta_i + c_i\phi) + J_{cc}(\theta_i + c_i\phi) < J(\theta_i) + J_{cc}(\theta_i)$ **do**:
                    $\theta_i \leftarrow \theta_i + c_i\phi$
        delete worst $S/2$ and reproduce best $S/2$
    for $i \leftarrow 1 \ldots S$ **do**:
        if $\epsilon \sim \mathcal{U}(0,1) < p_{ed}$ **then**:
            $\theta_i \sim d(\theta)$

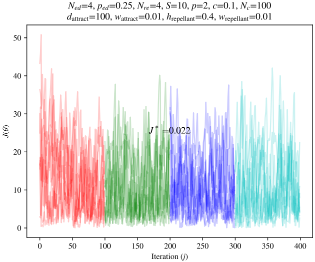
- N_{ed}: number of elimination-dispersal events
- p_{ed}: probabilty of a single elimination-dispersal event
- $d(\theta)$: initial distribution of θ

Does Elimination-Dispersal Help?









Caveats

- This is only a single application:
  - A single highly nonconvex function
  - Should try out other loss functions and applications (see post-presentation slides)

Caveats

- This is only a single application:
  - A single highly nonconvex function
  - Should try out other loss functions and applications (see post-presentation slides)
- Chose a single seed arbitrarily to run experiments:
  - Results could vary if a different seed was chosen

Caveats

- This is only a single application:
  - A single highly nonconvex function
  - Should try out other loss functions and applications (see post-presentation slides)
- Chose a single seed arbitrarily to run experiments:
  - Results could vary if a different seed was chosen
- Hyperparameters were chosen by trying a few random combinations
  - Tried to avoid "overfitting" hyperparameters (abuse of terminology)

Is this a good optimization algorithm?

- Final algorithm checks at least

$$N_{ed} \times N_{re} \times N_c \times S$$