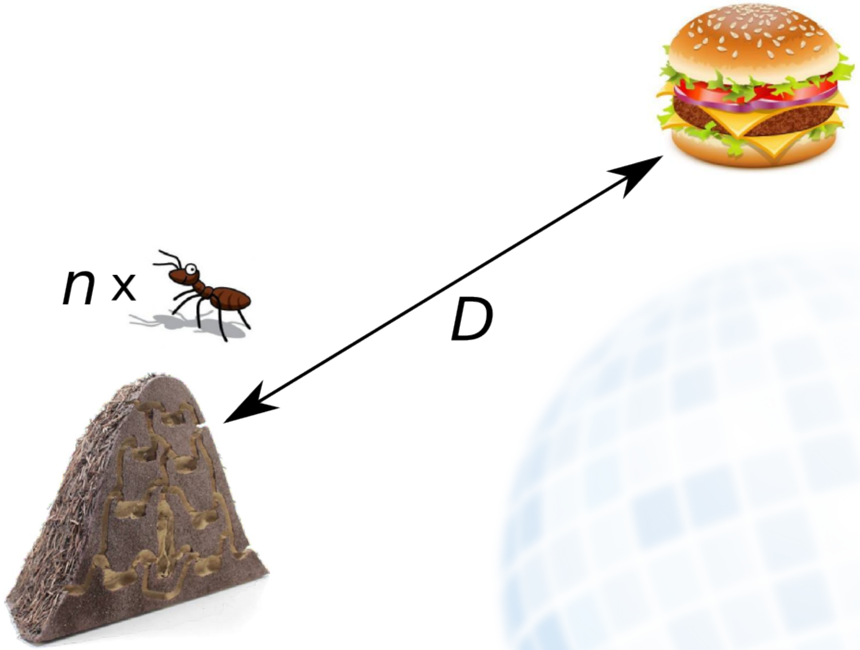
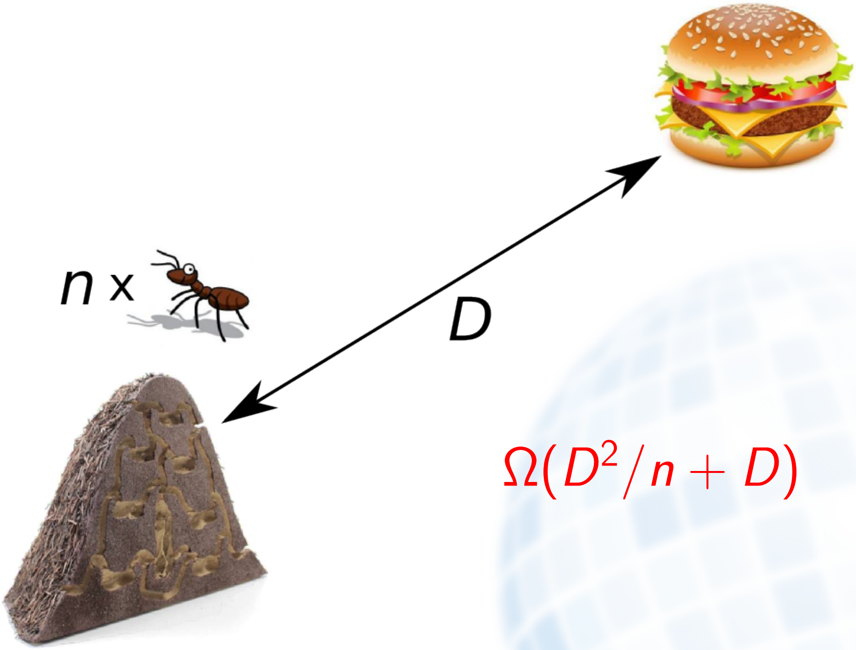
Solving the ANTS Problem with Asynchronous Finite State Machines

Yuval Emek, ***Tobias Langner***, *Jara Uitto*, *Roger Wattenhofer*

n x 



n x
D


n x
D
$\Omega(D^2/n + D)$

Previous Work

- ANTS problem (Ants Nearby Treasure Search) introduced by Feinerman, Korman, Lotker, Sereni [PODC 2012]

=

Previous Work

- ANTS problem (Ants Nearby Treasure Search) introduced by Feinerman, Korman, Lotker, Sereni [PODC 2012]

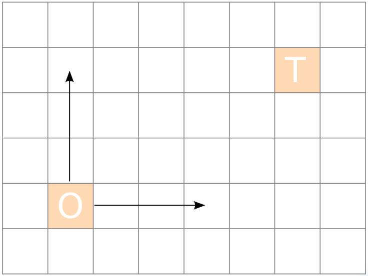
 $=$ 

- Treasure located in **optimal** time $\mathcal{O}(D^2/n + D)$

Motivation

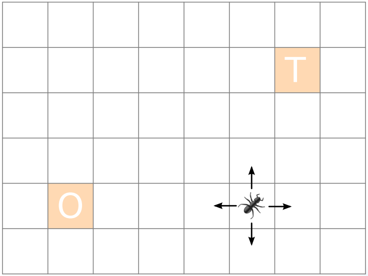
- "They operate without any central control. Their collective behavior arises from local interactions."
- Prime example of **real-world** distributed algorithms
- Deeper understanding may help computer science and biology

Model



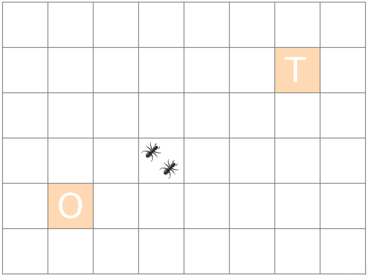
- Infinite integer grid with **origin** and **treasure** in (Manhattan-) distance D
- Ants controlled by the same **randomized finite automaton**
- Execution in **asynchronous environment**
- **Goal:** Starting at origin, find treasure fast

Model



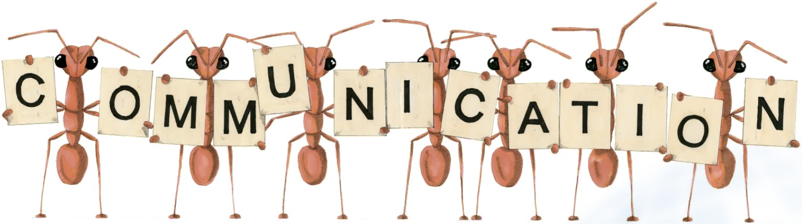
- In each step, ant can move **one cell** N, E, S, W or stay

Model



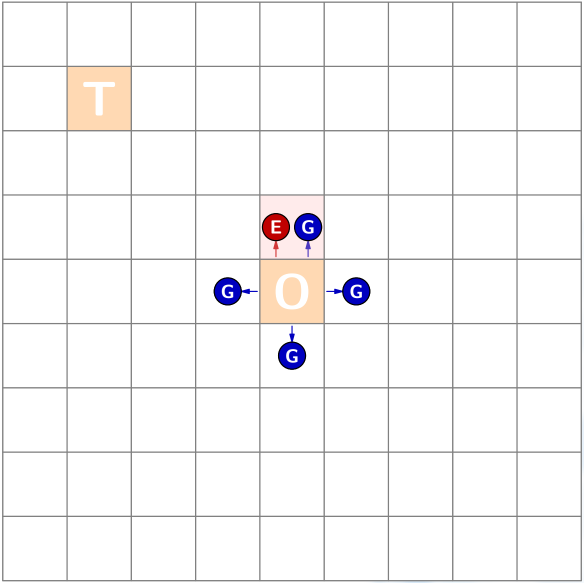
- **Communication** within cells

Model

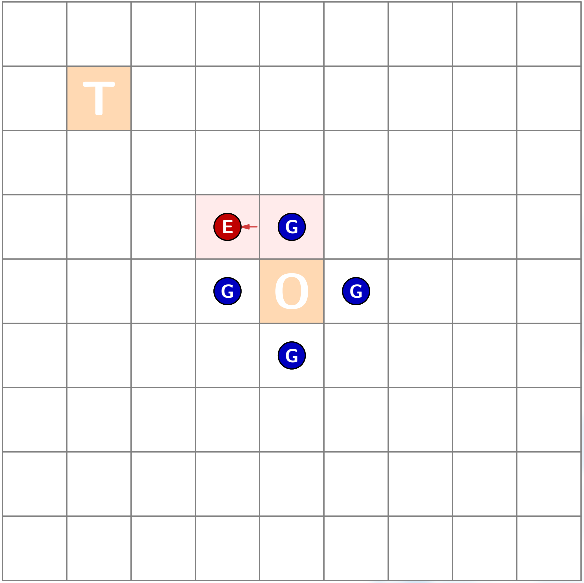


- For each **state**: Is there an ant with this state?
- $\Rightarrow$ **Finite** message size

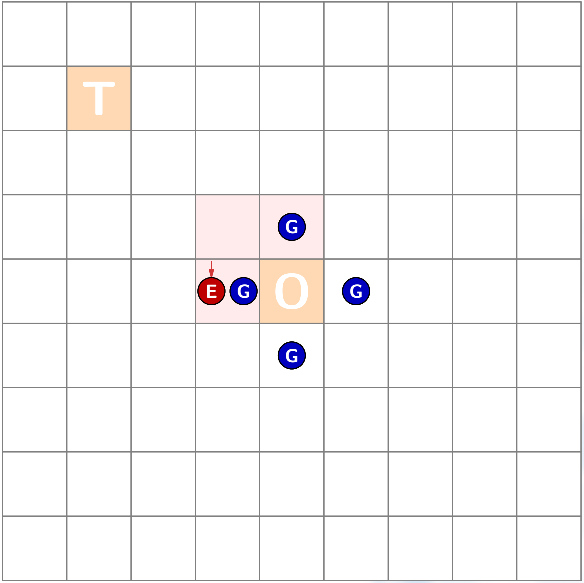
Diamond Search



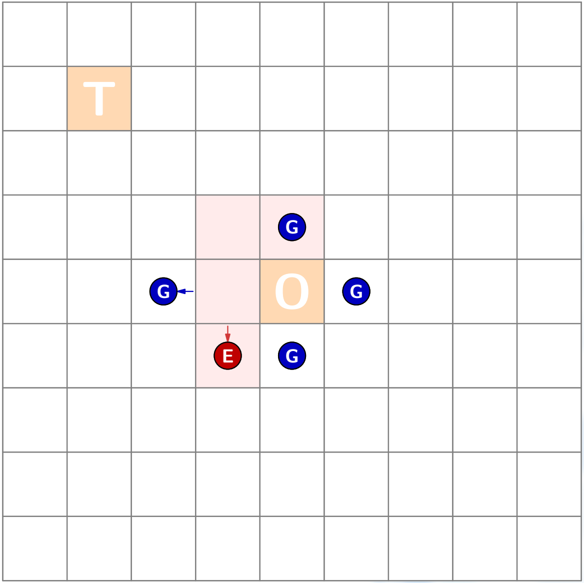
Diamond Search



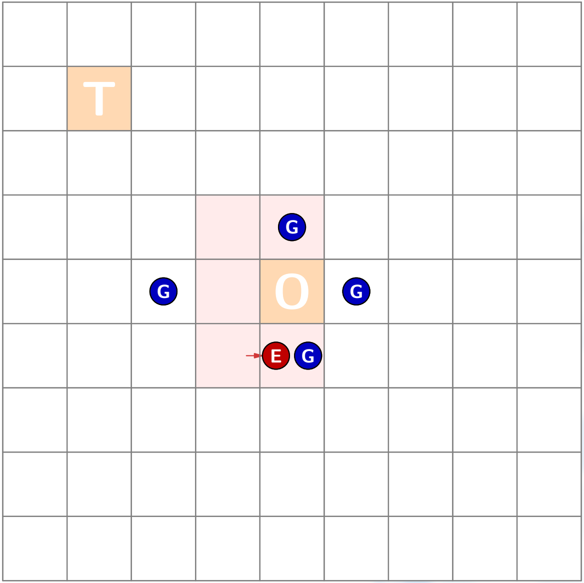
Diamond Search



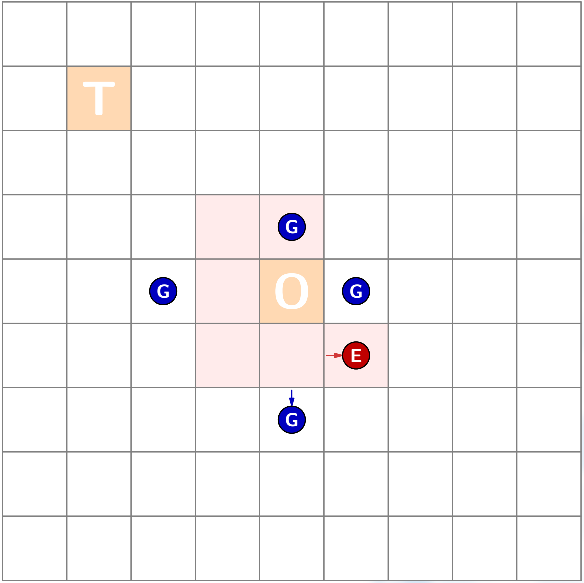
Diamond Search



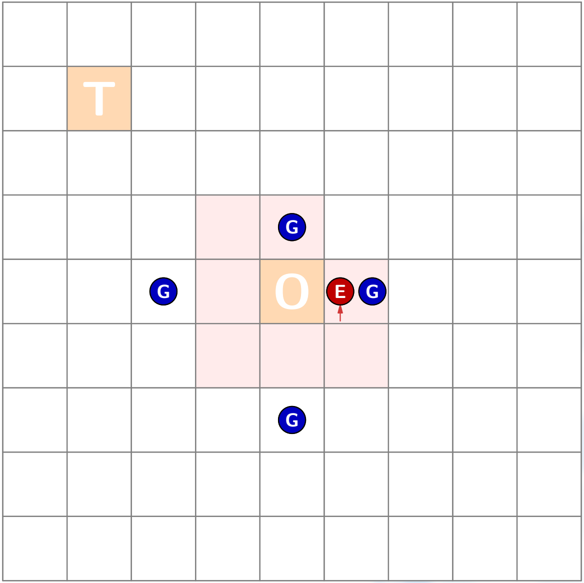
Diamond Search



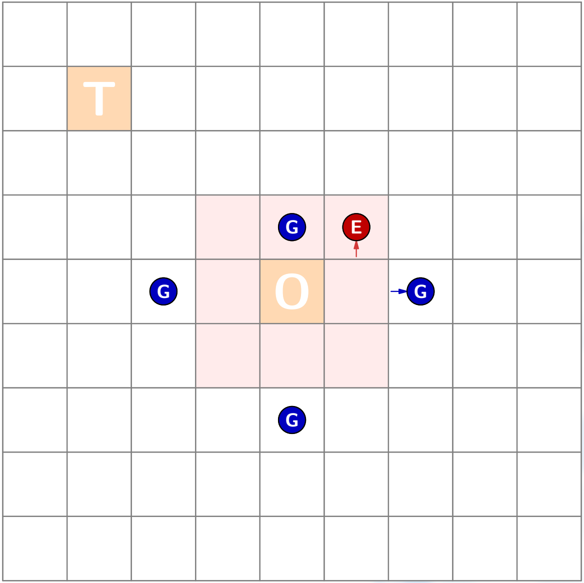
Diamond Search



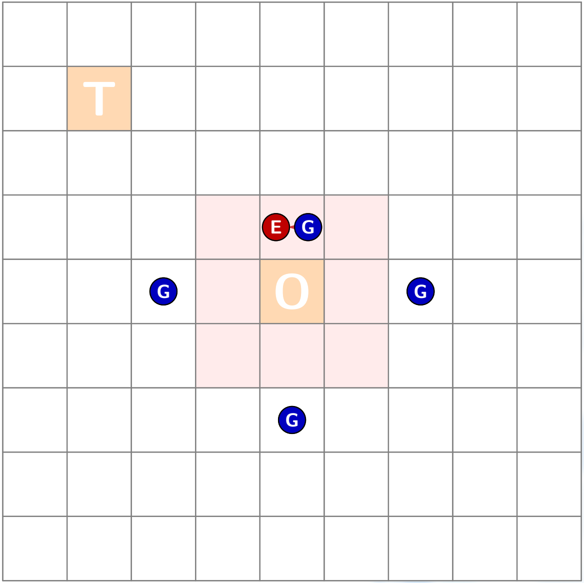
Diamond Search



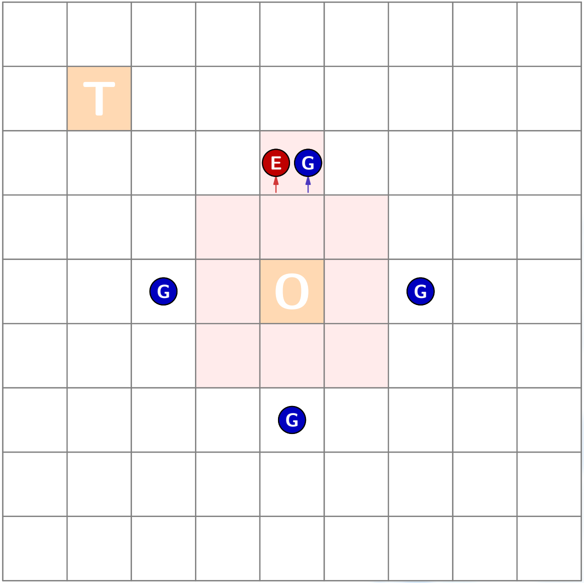
Diamond Search



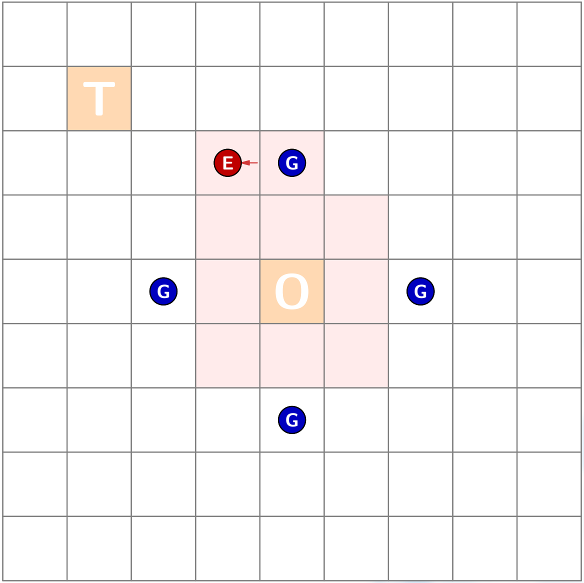
Diamond Search



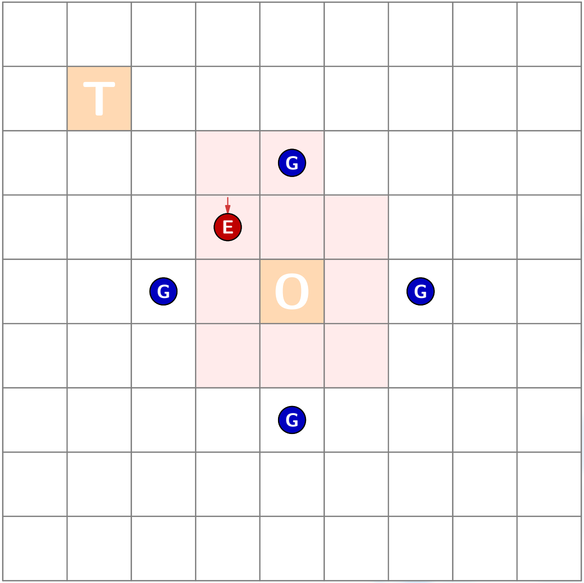
Diamond Search



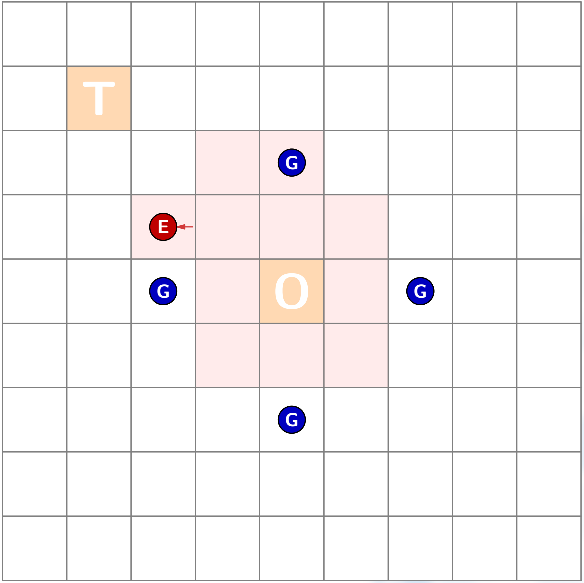
Diamond Search



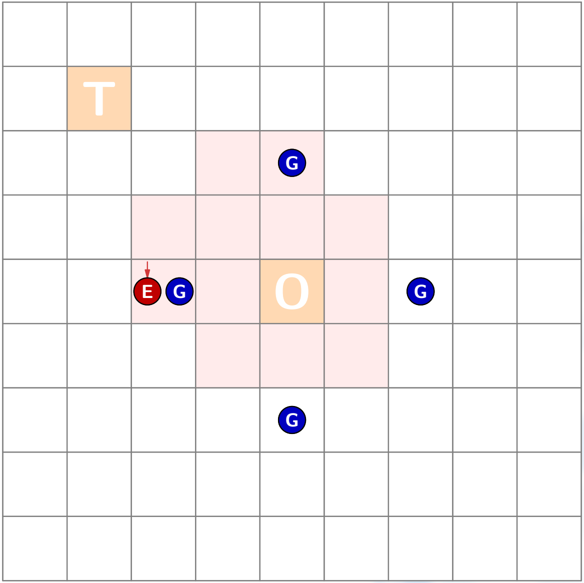
Diamond Search



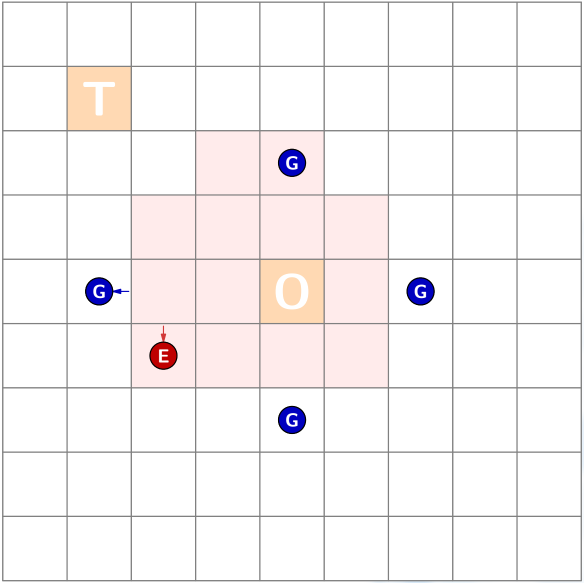
Diamond Search



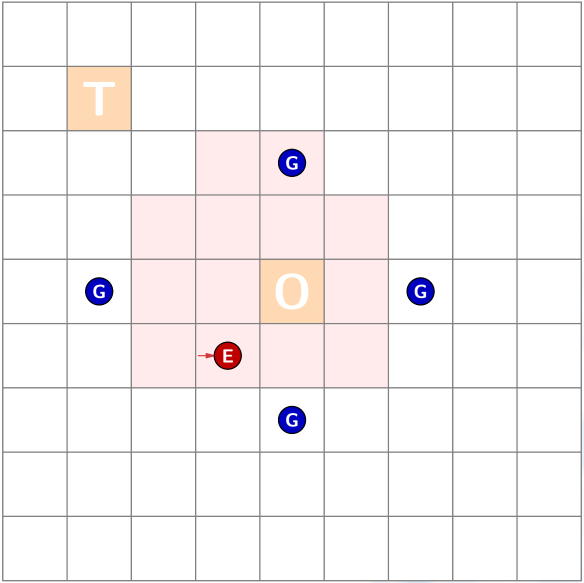
Diamond Search



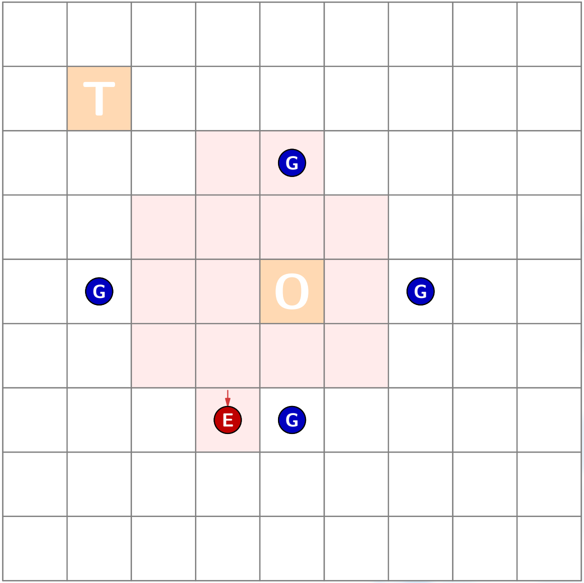
Diamond Search



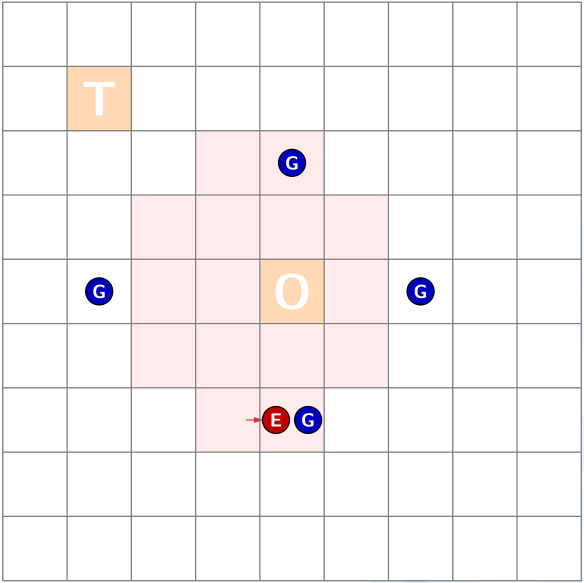
Diamond Search



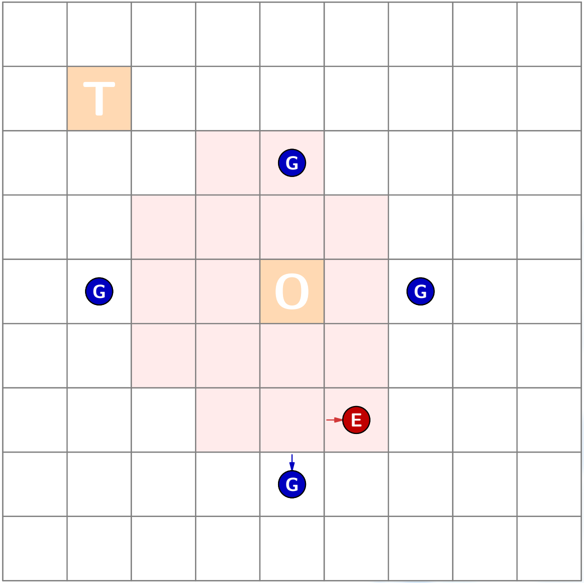
Diamond Search



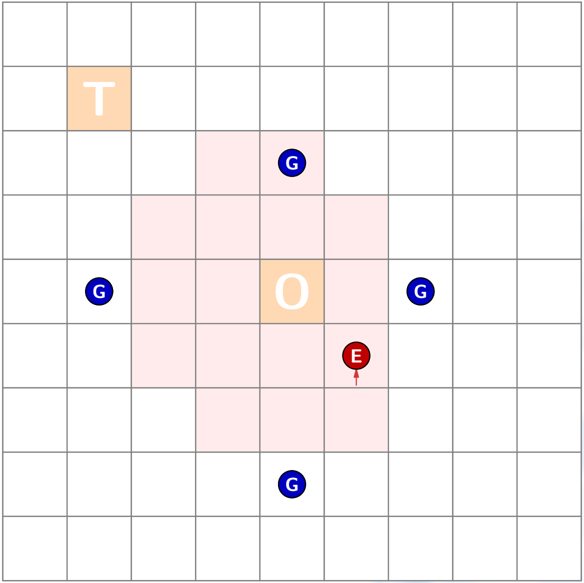
Diamond Search



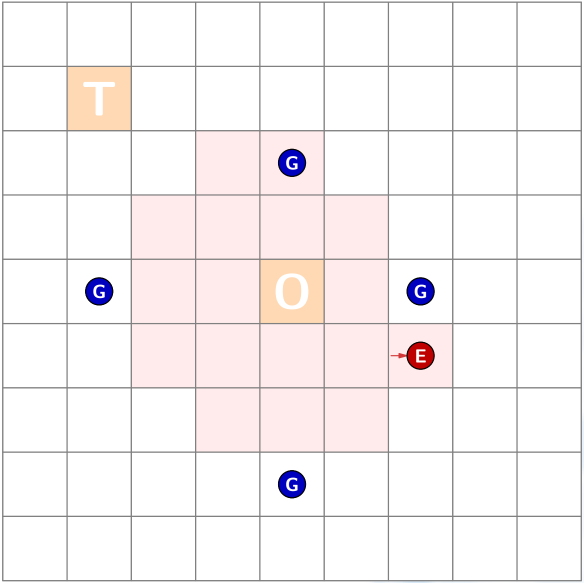
Diamond Search



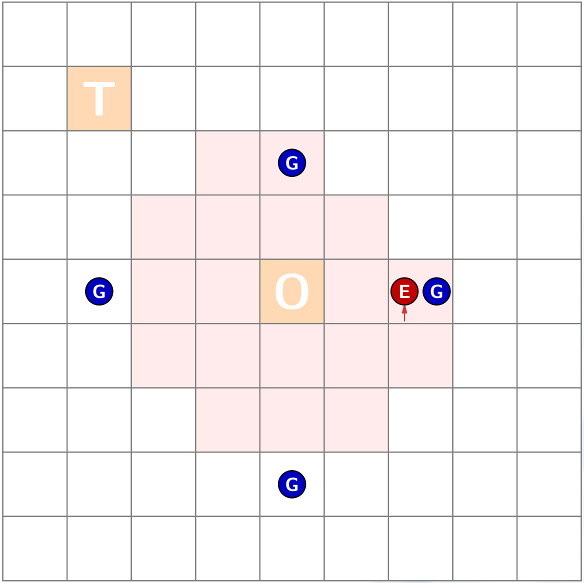
Diamond Search



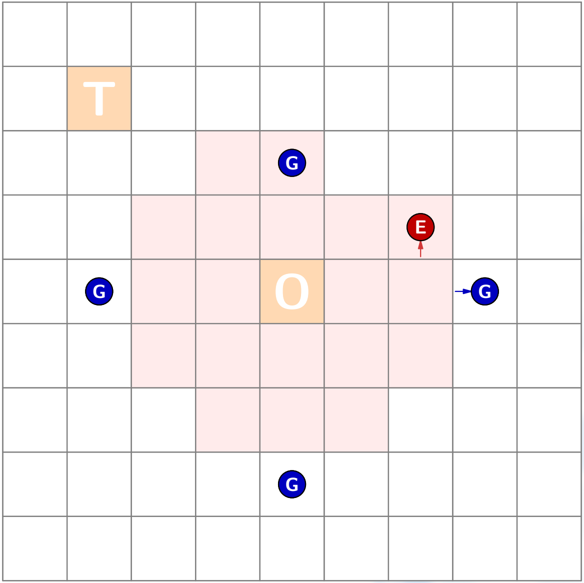
Diamond Search



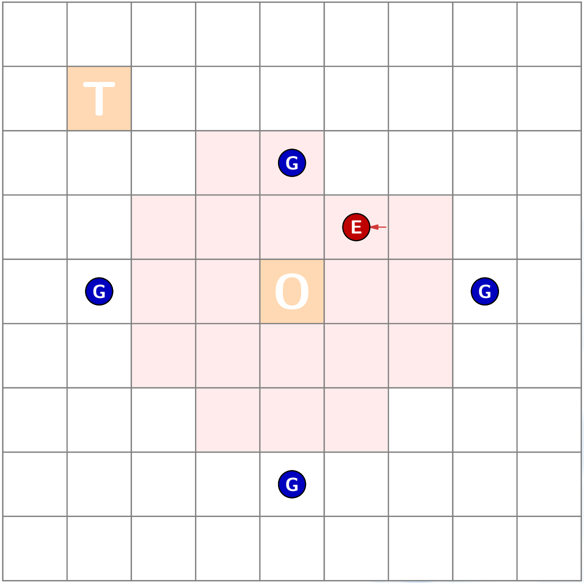
Diamond Search



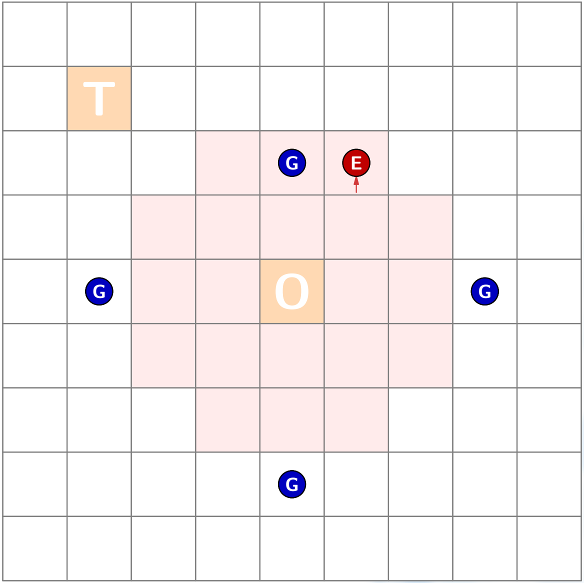
Diamond Search



Diamond Search



Diamond Search



Diamond Search – Runtime

- $\mathcal{O}(D^2)$ cells within distance D
- New cell explored every **constant number of steps**
- Runtime: $\mathcal{O}(D^2)$
- How can we **parallelize** it?

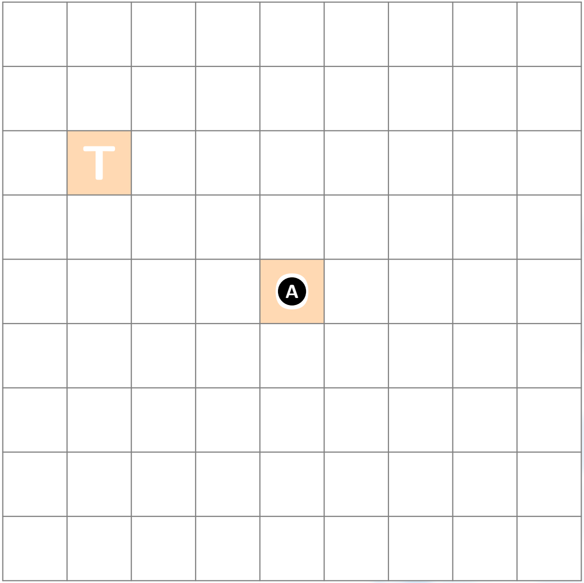
Parallel Diamond Search

- Simple idea: **Multiple search teams** search in parallel
- **Emit new team** as long as still ants available in origin
- Ensure "organized" overtaking

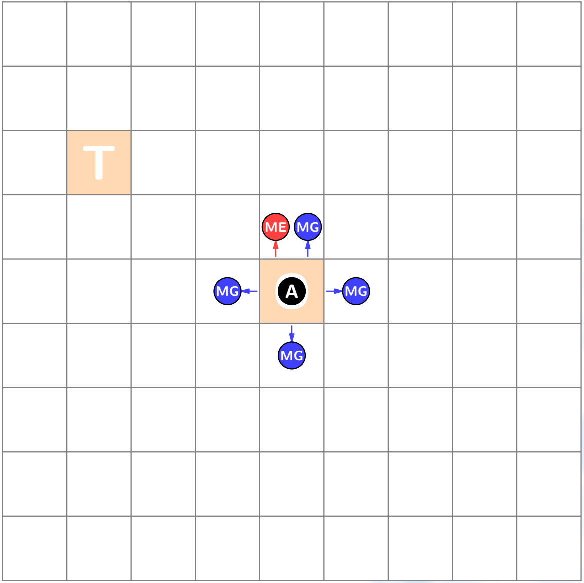
Parallel Diamond Search

- Search teams **stick together**
- Two separate stages
  - **Initialization**
  - **Search**

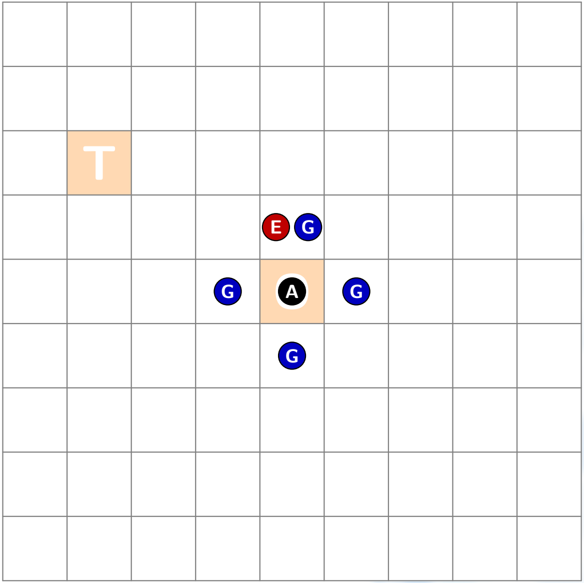
Parallel Diamond Search – Execution



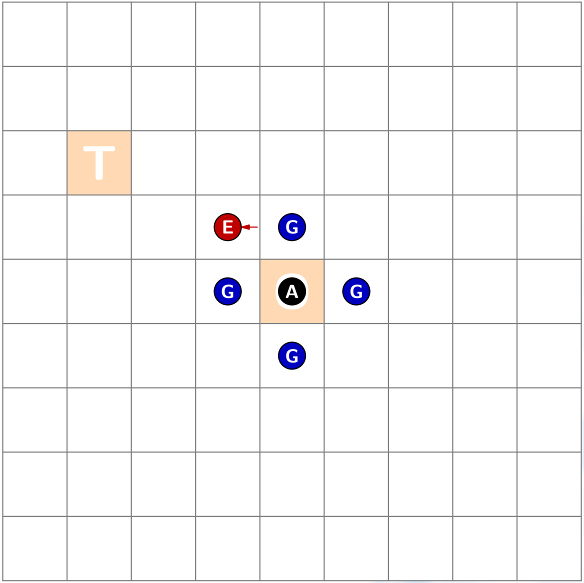
Parallel Diamond Search – Execution



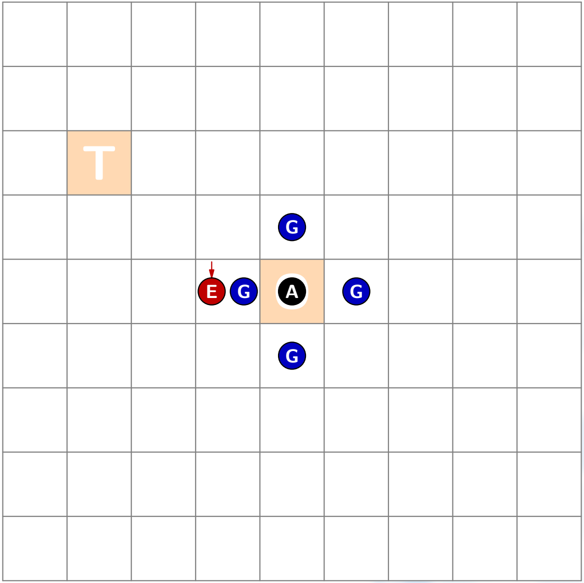
Parallel Diamond Search – Execution



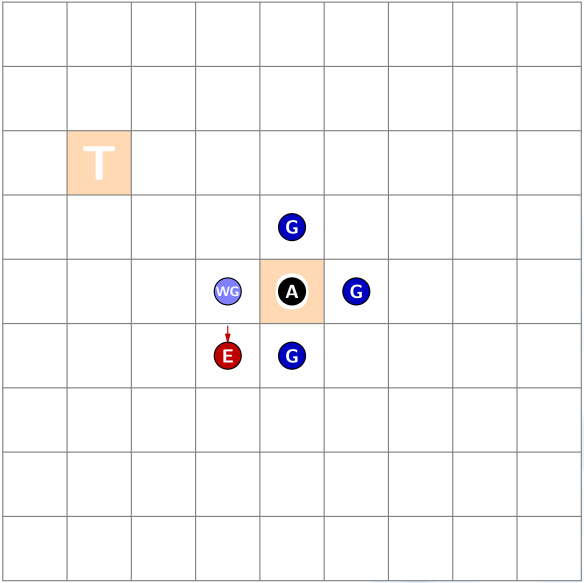
Parallel Diamond Search – Execution



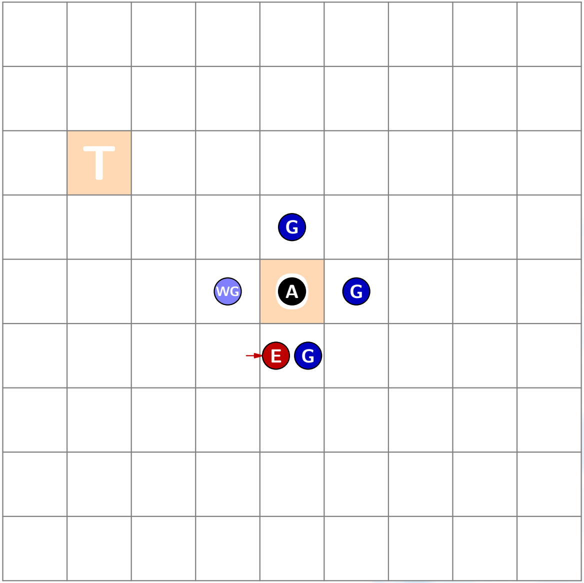
Parallel Diamond Search – Execution



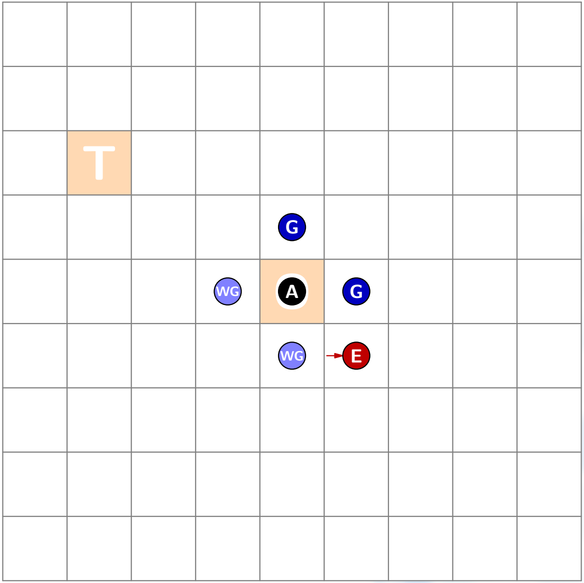
Parallel Diamond Search – Execution



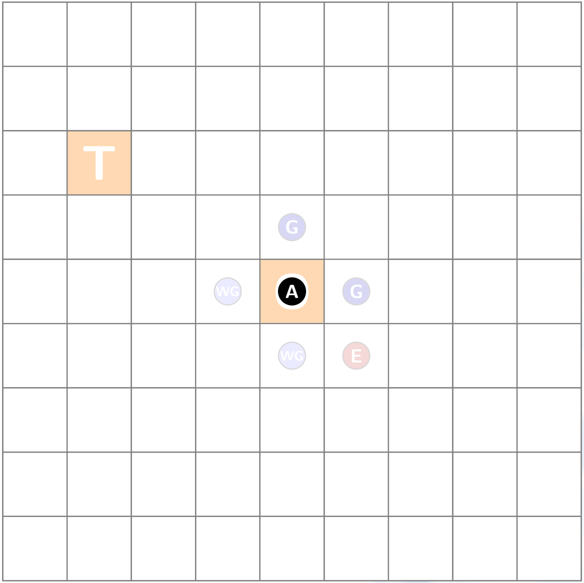
Parallel Diamond Search – Execution



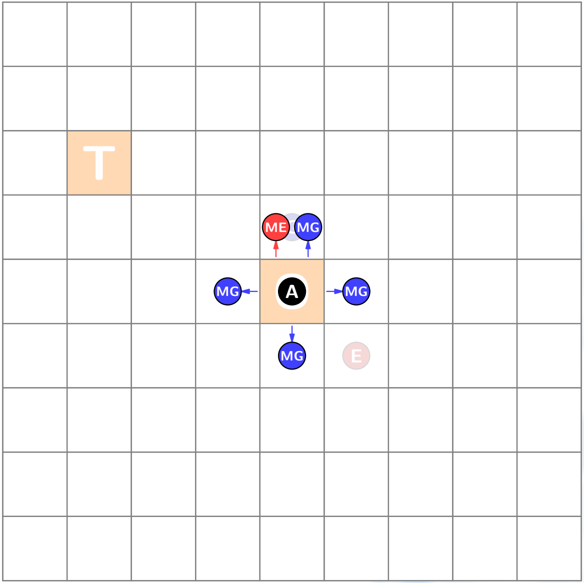
Parallel Diamond Search – Execution



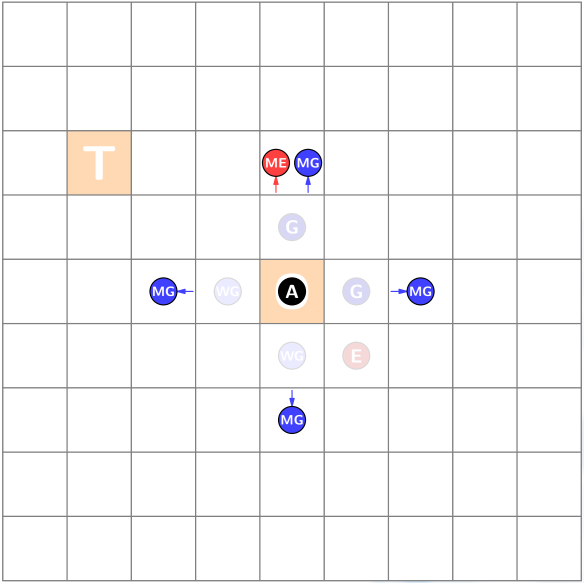
Parallel Diamond Search – Execution



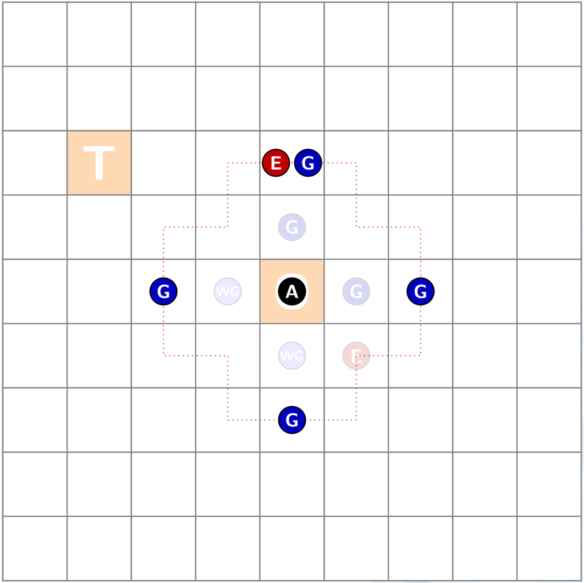
Parallel Diamond Search – Execution



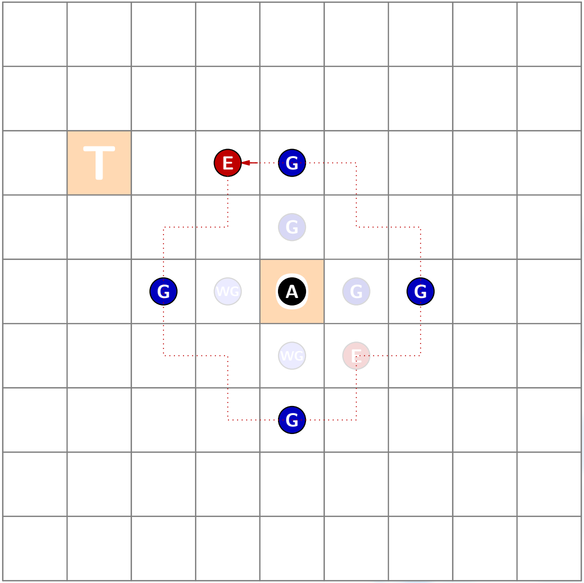
Parallel Diamond Search – Execution



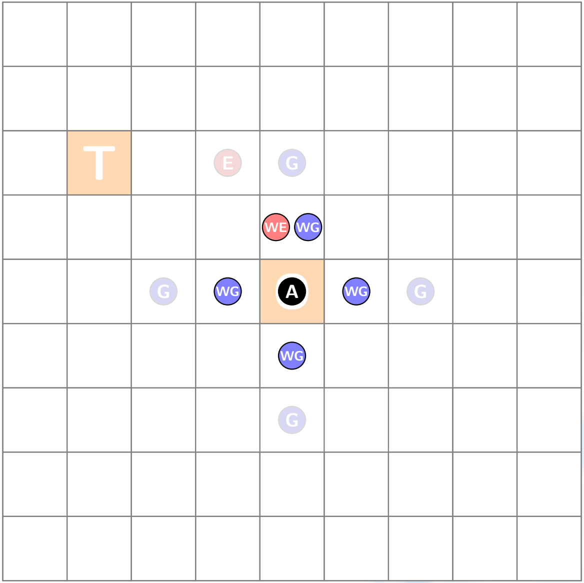
Parallel Diamond Search – Execution



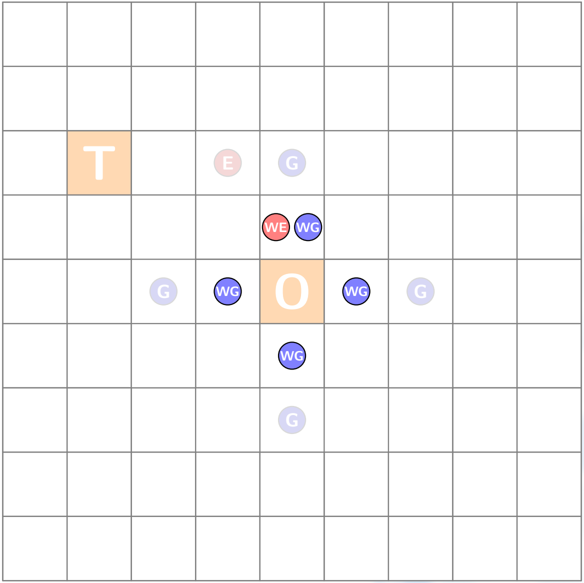
Parallel Diamond Search – Execution



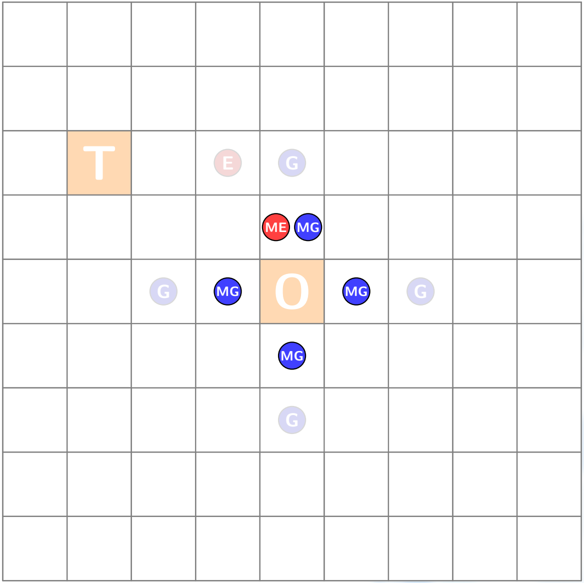
Parallel Diamond Search – Execution



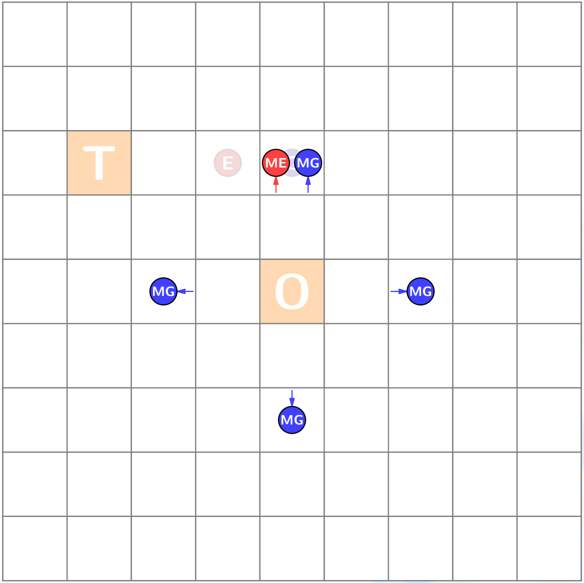
Parallel Diamond Search – Execution



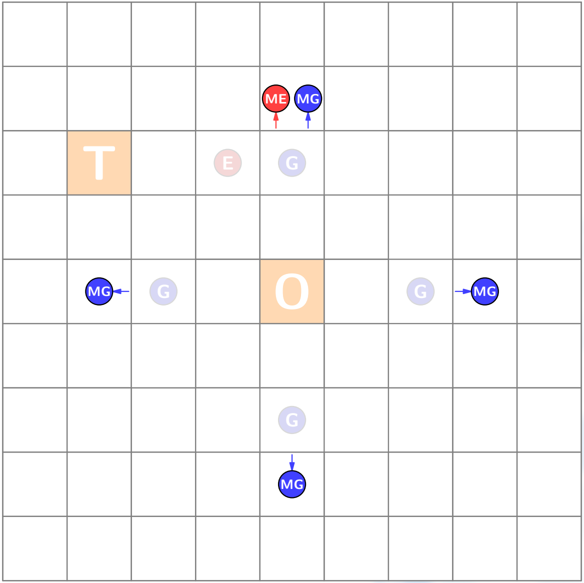
Parallel Diamond Search – Execution



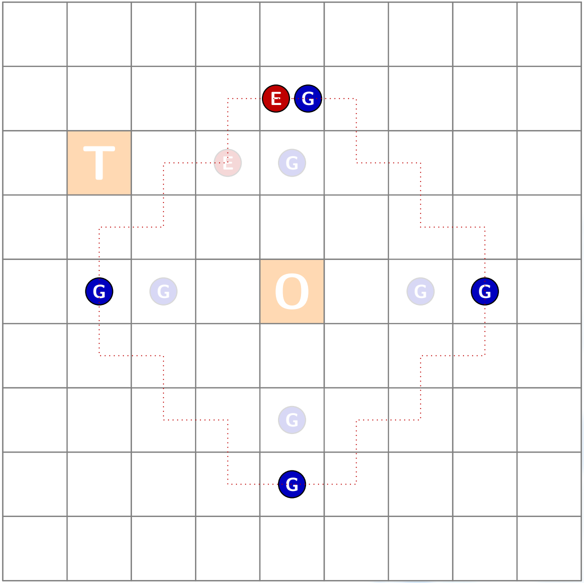
Parallel Diamond Search – Execution



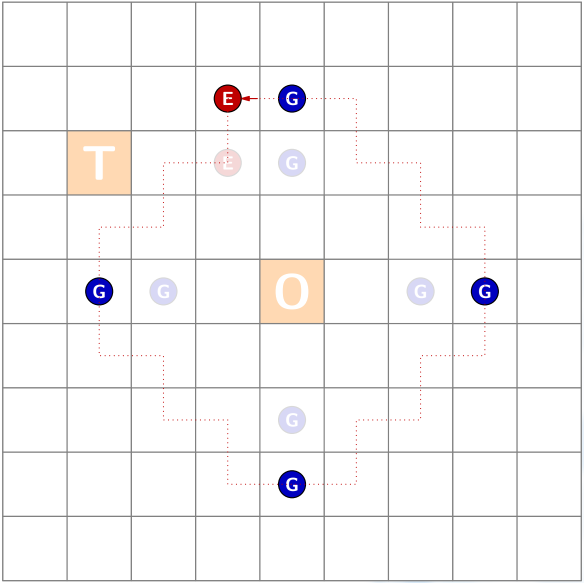
Parallel Diamond Search – Execution



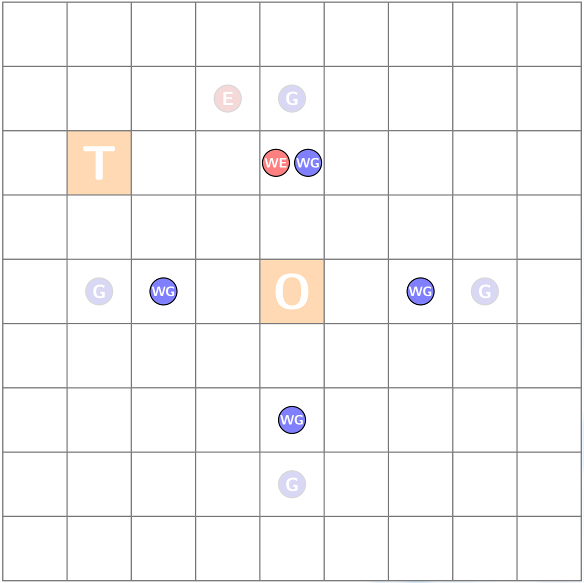
Parallel Diamond Search – Execution



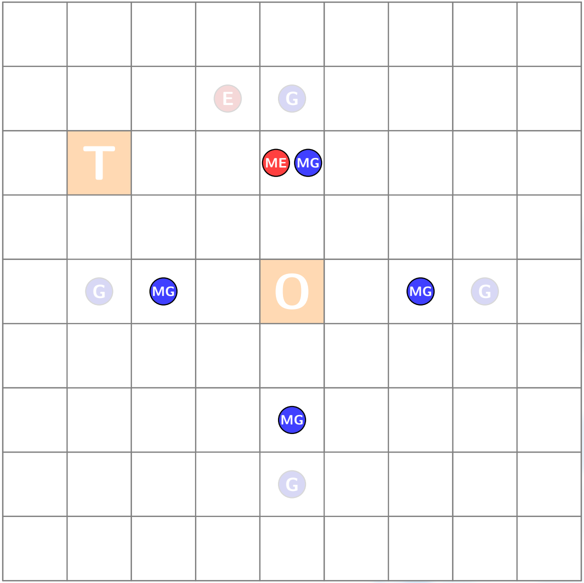
Parallel Diamond Search – Execution



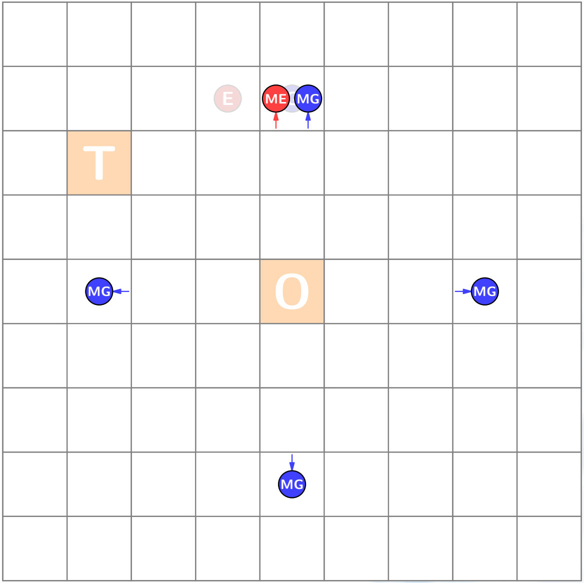
Parallel Diamond Search – Execution



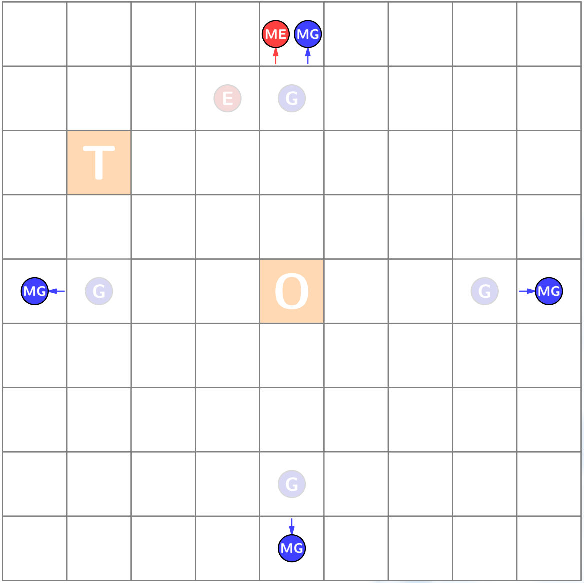
Parallel Diamond Search – Execution



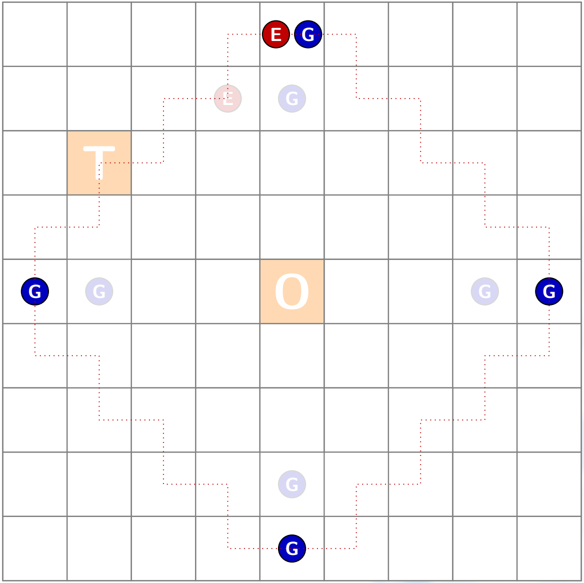
Parallel Diamond Search – Execution



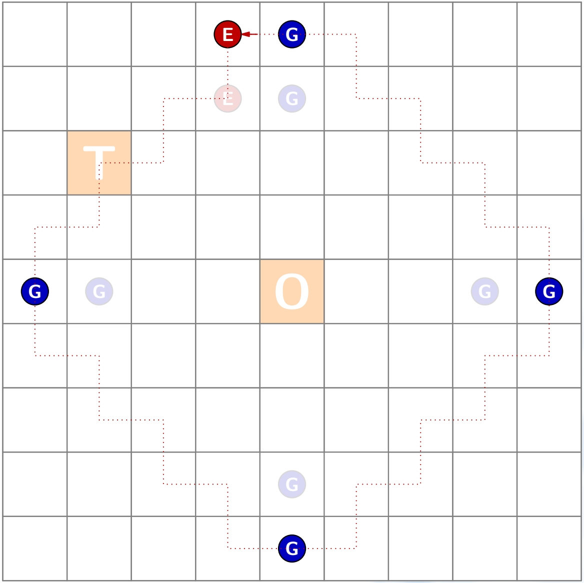
Parallel Diamond Search – Execution



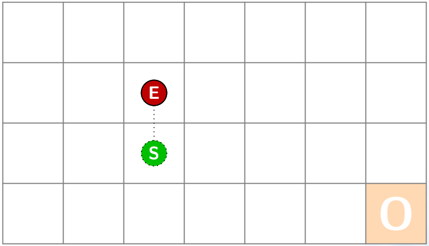
Parallel Diamond Search – Execution



Parallel Diamond Search – Execution

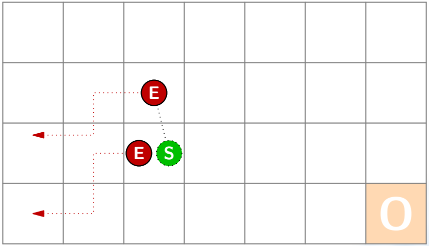


Handling Asynchrony



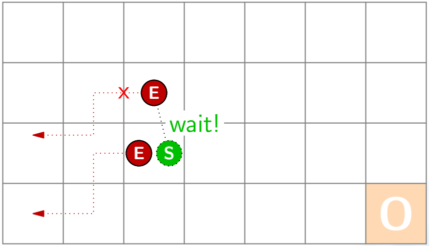
- Each ant employs a **Scout**

Handling Asynchrony



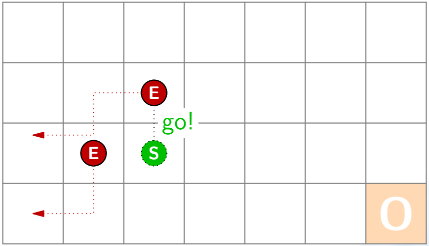
- Explorers do not **overtake**

Handling Asynchrony



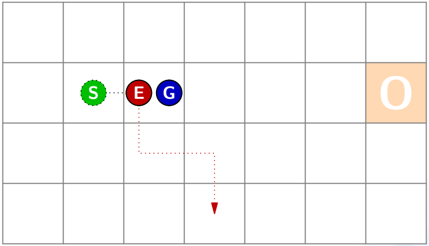
- Explorers do not **overtake**

Handling Asynchrony



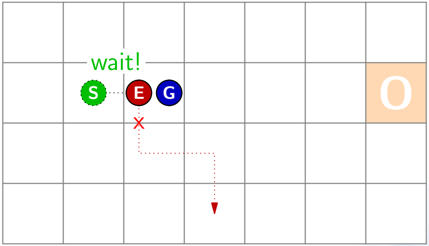
- Explorers do not **overtake**

Handling Asynchrony



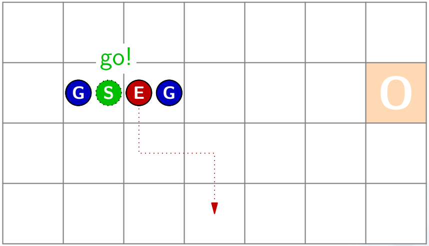
- Explorer **waits** for next guide

Handling Asynchrony



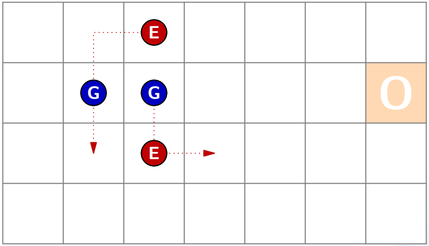
- Explorer **waits** for next guide

Handling Asynchrony



- Explorer **waits** for next guide

Handling Asynchrony



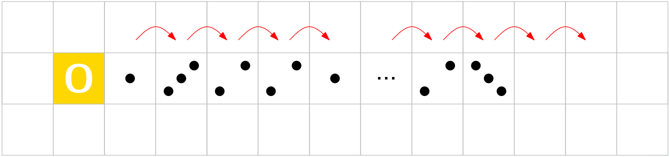
- Every explorer finds **guide** on axis

Runtime Analysis

Essential puzzle pieces:

- **Synchronous** schedule: no explorer is **delayed**
- **Synchronous** schedule: treasure found in time $\mathcal{O}(D^2/n + D)$.
- Synchronous schedule is **worst-case** schedule

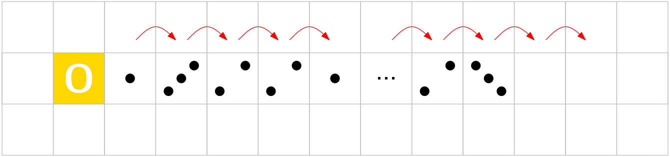
Emitting Search Teams



How can we repeatedly emit a new search team of ten ants?

- **Spread** ants along the east axis

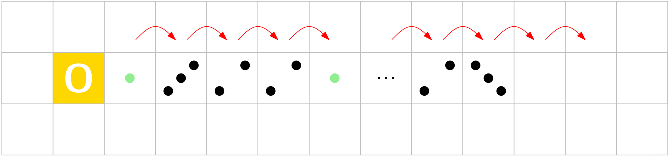
Emitting Search Teams



How can we repeatedly emit a new search team of ten ants?

- **Spread** ants along the east axis
- Each ant throws a **coin**

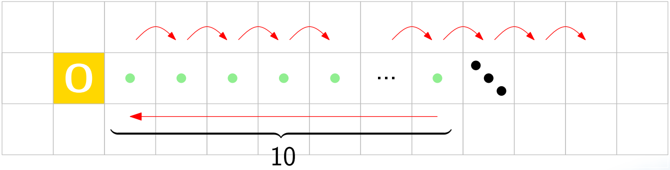
Emitting Search Teams



How can we repeatedly emit a new search team of ten ants?

- **Spread** ants along the east axis
- Each ant throws a **coin**
- When a cell contains **exactly one ant** $\Rightarrow$ **Ready**!

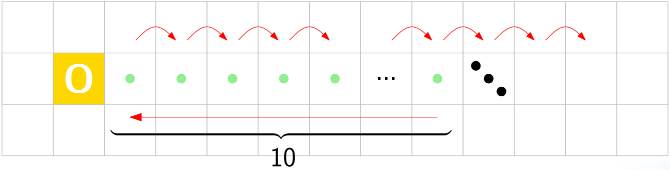
Emitting Search Teams



How can we repeatedly emit a new search team of ten ants?

- **Spread** ants along the east axis
- Each ant throws a **coin**
- When a cell contains **exactly one ant** $\Rightarrow$ **Ready**!
- Ten subsequent cells are **ready**? $\Rightarrow$ **Collect team**!

Emitting Search Teams



How can we repeatedly emit a new search team of ten ants?

- **Spread** ants along the east axis
- Each ant throws a **coin**
- When a cell contains **exactly one ant** $\Rightarrow$ **Ready**!
- Ten subsequent cells are **ready**? $\Rightarrow$ **Collect team**!
- After time $\mathcal{O}(\log n)$, a new team can be emitted within a constant amount of time

Putting Everything Together

- Search team emission works after time $\mathcal{O}(\log n)$

Putting Everything Together

- Search team emission works after time $\mathcal{O}(\log n)$
- **Then**, treasure is found in time $\mathcal{O}(D^2/n + D)$

Putting Everything Together

- Search team emission works after time $\mathcal{O}(\log n)$
- **Then**, treasure is found in time $\mathcal{O}(D^2/n + D)$
- Runtime: $\mathcal{O}(D^2/n + D + \log n)$

Putting Everything Together

- Search team emission works after time $\mathcal{O}(\log n)$
- **Then**, treasure is found in time $\mathcal{O}(D^2/n + D)$
- Runtime: $\mathcal{O}(D^2/n + D + \log n)$
    - About half of the ants perform **RectangleSearch**

Putting Everything Together

- Search team emission works after time $\mathcal{O}(\log n)$
- **Then**, treasure is found in time $\mathcal{O}(D^2/n + D)$
- Runtime: $\mathcal{O}(D^2/n + D + \log n)$
  - About half of the ants perform **RectangleSearch**
  - The other half performs **local random search** which locates treasure in time $\mathcal{O}(D)$ if $D \leq \log n$

Putting Everything Together

- Search team emission works after time $\mathcal{O}(\log n)$
- **Then**, treasure is found in time $\mathcal{O}(D^2/n + D)$
- Runtime: $\mathcal{O}(D^2/n + D + \log n)$
  - About half of the ants perform **RectangleSearch**
  - The other half performs **local random search** which locates treasure in time $\mathcal{O}(D)$ if $D \leq \log n$
- **Combined runtime**: $\mathcal{O}(D^2/n + D)$

Diamond Search in Real Life

Thanks!
Questions & Comments?