Metaheuristics and Cognitive Models for Autonomous Robot Navigation

Raj Korpan RAJ.M.KORPAN@GMAIL.COM

Department of Computer Science

The Graduate Center, CUNY

Abstract

Although autonomous mobile robots have become more commonplace, methods for them to navigate in the real-world are not yet perfected. This survey describes two heuristic-based approaches, metaheuristics and cognitive models, that have been used in this domain under separate but concurrent development. Metaheuristics are broadly-applicable, heuristic-based algorithms inspired by natural processes. They use one or more heuristic strategies to search for solutions. Alternatively, studies have shown that people adaptively use one or more strategies during navigation. Cognitive models are used to computationally encode human behaviors. This paper details both approaches, reviews the literature on their application to autonomous robot navigation, and discusses their advantages and disadvantages. Finally, to take advantage of algorithmic efficiency and human behaviors, the conclusion proposes the integration of these two approaches.

Keywords: Autonomous robot navigation, path planning, metaheuristics, cognitive models

Table of Contents

1. Introduction

Robots are increasingly prevalent in modern society, where they will eventually be expected to complete tasks autonomously. In particular, mobile robots must navigate in many settings, including social assistance, manufacturing, delivery, exploration, and search and rescue. Planning an optimal path in a non-trivial environment, however, is NP-hard. This paper reviews two significant areas of research in autonomous robot navigation: metaheuristics (high level heuristic techniques) and cognitive models (computational simulations of human behavior). Studies have shown that people use multiple heuristics and change their strategy across navigational tasks and within tasks. Much the way people adapt their strategy based on their circumstances, metaheuristics seek to use a heuristic strategy to solve problems. Although heuristic methods find solutions that may merely be good enough, for autonomous robot navigation such solutions must suffice. Both approaches provide viable solutions to the problem of autonomous robot navigation but no approach has emerged as uniformly superior.

This chapter formalizes artificial intelligence problems and their solutions. It begins with fundamental concepts and describes problems in the context of autonomous robot navigation. It then discusses challenges in autonomous robot navigation. Finally, it describes heuristics and metaheuristics.

1.1 Problem Formulation

In Artificial Intelligence (AI), an *agent* perceives its environment through *sensors* and acts upon that environment with *actuators* (Russell and Norvig, 2009). An *embodied* agent has a physical body that interacts with the environment through that body. A *robot* is an artificial, embodied, mechanical agent. Examples of robot sensors include

cameras and lasers; an example of a robot actuator is a motor. A *mobile* robot can move through its environment. At any instant, a mobile robot has a *pose* $\langle x, y, \theta \rangle$ in its environment, where $\langle x, y \rangle$ is its position and θ is its orientation with respect to some coordinate system. An *autonomous* robot acts without human intervention.

An agent addresses a *problem*, defined by the tuple $\mathcal{P} = \langle S, I, A, G \rangle$ where

- S is a set of states that represent an instance of the environment. Each state $s \in S$ is represented as a set of features
- $I \subseteq S$ is a set of initial states
- A is a set of possible actions from which the agent can select
- $G(s)$ is a Boolean goal test

An agent uses its actuators to perform an *action* $a \in A$ that is intended to take the agent from state s_i to the next state s_j. If an action fails or is an intentional pause, then s_i may be the same as s_j. A *problem class* is a set of problems that share some characteristic. A *problem domain* is a set of related problem classes.

Autonomous robot navigation is a problem domain in which autonomous mobile robots move through an environment from one location to another. This paper takes autonomous robot navigation as a running example of an AI problem. Research on autonomous robot navigation is motivated by the deployment of robots in nearly every aspect of society. Robots must now travel and complete tasks independently in complex environments, such as factory and warehouse floors, offices, and homes, often in the company of people, other robots, and obstacles. The states $s = \langle x, y, \theta \rangle$ in autonomous robot navigation represent the possible poses of the robot in its environment. An initial state is the pose at which the robot begins, such as the position of a charging station or the entrance to a building, and the direction the robot faces

in that position. Each action in A is either intended to change the robot's pose (e.g., a forward movement or a turn) or is an intentional pause which does not change the pose. Finally, the goal test G returns true only if the robot's current position is the target's location.

Given a problem $\mathcal{P} = \langle S, I, A, G \rangle$, a *path* p is a finite ordered sequence of interleaved states and actions $\langle s_1, a_1, s_2, a_2, s_3, \ldots, s_{r-1}, a_{r-1}, s_r \rangle$. The agent attempts to find a *solution*, a finite path $p = \langle s_1, a_1, s_2, a_2, s_3, \ldots, s_{r-1}, a_{r-1}, s_r \rangle$ from an initial state to a goal state, that is, $s_1 \in I$ and $G(s_r) = True$. A *plan* is a path whose action sequence is selected and can be proved to be a solution before it is executed. *Actuator error* occurs when actuators do not execute an action exactly as intended. As a result of actuator error, although an agent intends to follow a plan, it may deviate so that the path actually travelled differs from the plan.

The *search space* H for a problem $\mathcal{P}$ is the set of all paths that start at an initial state. Many paths in the search space do not reach a goal state. For example, in autonomous robot navigation, a path p is a sequence of poses interleaved with actions (e.g., s_1 = initial pose, a_1 = move forward two feet, s_2 = next pose, a_2 = turn left 90°, s_3 = next pose, a_3 = move forward five feet). A solution for the robot is a path that starts at an initial location and ends at a target. The search space could contain infinitely many paths that the robot could take from an initial pose.

A metric defines the *step cost* c to take an action a in state s. The step cost may be uniform across all state-action pairs, or may be defined in the context of the problem class. Examples of metrics for a mobile robot include elapsed time and battery usage. The step cost c of an action in autonomous robot navigation can be measured as the amount of energy consumed, the time taken, or the distance traveled.

The *path cost* $C(p)$ of path p is the sum $C(p) = \sum_i c_i$ of the step costs of all the actions along p. For example, the path cost $C(p)$ in autonomous robot navigation

could measure the total battery usage or the distance traveled for the entire path. An *optimal solution* o is a solution in the search space with minimum path cost: $o = \arg\min_{p \in H} C(p)$. An optimal solution in autonomous robot navigation is defined by its metric, for example, as the fastest or the shortest path from the initial position to the target. A *satisfactory solution* is a solution that is good enough with respect to a domain-specific criterion, and thus is less likely to require prohibitive computational resources (Poole and Mackworth, 2010). Satisfactory solutions are usually sub-optimal. A satisfactory solution in autonomous robot navigation would be a sufficiently short path or a sufficiently fast one.

Search is a process that explores a problem's search space to find a solution. In an *optimization problem*, search seeks an optimal solution. For NP-hard problems with sufficiently large search spaces, however, it is often impractical or impossible to compute an optimal solution. A particular challenge to heuristics for optimization is *premature convergence*, which is when the search stops at a sub-optimal solution. Examples of optimization problems in graph search include the traveling salesperson problem (Dantzig et al., 1954) and the minimum spanning tree problem (Graham and Hell, 1985).

Path planning for autonomous robot navigation is the search for a plan that minimizes travel time, travel distance, or resource consumption (Fong et al., 2015). Because path planning is NP-hard, the discovery of optimal paths is intractable in any non-trivial environment (Canny, 1988). Thus, for many important autonomous robot navigation tasks that involve path planning, a satisfactory solution must suffice. The next section discusses additional challenges in autonomous robot navigation.

1.2 Challenges in Autonomous Robot Navigation

Autonomous robot navigation faces several challenges beyond optimal path planning: localization, mapping, obstacle avoidance, and motion control. *Localization* requires an autonomous robot to detect its current pose. When a robot cannot localize accurately in its environment, it is *lost*. *Mapping* requires the robot to construct a map of its environment if it is unknown. *Obstacle avoidance* requires the robot to move through its environment without collisions. *Motion control* requires the robot to manipulate its actuators to perform intended actions despite actuator error.

Rather than one method to address all these challenges, most approaches tackle each of them separately. An exception to this is *simultaneous localization and mapping* (SLAM), a methodology in which the robot constructs a map of an unknown environment and simultaneously keeps track of its position within that map. Popular SLAM approaches use Bayesian probabilistic techniques, such as a Rao-Blackwellized particle filter or extended Kalman filter, to construct a probabilistic representation of the environment (Durrant-Whyte and Bailey, 2006). After the robot has mapped its environment (whether through SLAM or another method), a graph search algorithm is typically used for path planning. Examples of graph search algorithms include depth-first search, breadth-first search, Dijkstra's algorithm, A* search (Korf, 2014), D* search (Stentz, 1994), D* Lite (Koenig and Likhachev, 2002), LPA* (Koenig et al., 2004), and MPGAA* (Hernández et al., 2015).

The state-of-the-art for autonomous robot navigation is probabilistic SLAM for mapping and localization, and A* search for path planning. A* search, however, faces several situations in which other methods may be better. A* is only viable if a map of the environment is available, but the robot may not have time to create a map. Even if a map is available, in sufficiently complex problem domains A* search may

not find a solution in a reasonable amount of time. Although A* search provides optimal solutions, other methods may provide satisfactory solutions faster than A*, and, as previously described, satisfactory solutions must suffice in many autonomous robot navigation applications. Finally, A* search has a problem-specific component (its admissible heuristic) that must be carefully selected to ensure the optimality of the search but this component is not always easily obtained. Other approaches avoid this issue with general problem-independent strategies.

Additionally, although probabilistic SLAM with graph search is commonly used for path planning, there is no conclusive evidence that this approach is superior to other approaches because autonomous robot navigation lacks a standard testbed and standard performance metrics. This makes it difficult to compare results from different systems, especially when they make different assumptions about the environment. While path cost, computation time, and algorithmic complexity are appropriate performance metrics for path planning, no one metric is consistently used in the literature. As a result, researchers have proposed many alternatives to SLAM with graph search, including the approaches discussed in this paper.

Other challenges for autonomous robot navigation stem from the nature of the robot's environment. An agent's environment may be categorized by its observability. In a *fully observable* environment an agent's sensors view the complete state of the environment at each point in time. An environment is *partially observable* if parts of the environment are not detected by the sensors or if the sensors are noisy or inaccurate. Finally, an *unobservable* environment cannot be perceived by the agent.

Another way to characterize an environment is as *single agent* or *multiagent*. In multiagent environments, an agent must consider whether the other agents are *competitive* (work against each other), *cooperative* (work together), or neither (work independently). Path planning for multiple robots presents additional challenges,

such as whether to minimize individual path length versus total path length, and how to prevent collisions between robots. Another issue for multi-robot path planning is whether to consider each robot's path as a separate path planning problem or as one large optimization problem. An algorithm that considers the paths separately may be faster but may also have to address collisions between robots afterwards with path corrections or real-time obstacle avoidance.

A *static* environment does not change while an agent is deciding which action to take whereas a *dynamic* environment changes over time. In a dynamic environment, obstacles, other agents, and the structure of the environment itself can all move or change. Path planning in a dynamic environment is more difficult because the agent has to contend with changes in the environment that may make a once-satisfactory plan infeasible. A *continuous* environment has an infinite number of states; a *discrete* environment has a finite number of states. To create a discrete environment, the continuous dimensions of an environment can be discretized into evenly spaced intervals. For example, the states $s = \langle x, y, \theta \rangle$ in autonomous robot navigation are infinite since $(x, y) \in \mathbb{R}^2$ and $\theta \in [0, 360]$, but these can be discretized by binning the values to integers in a limited range. A discrete environment is typically represented as a two-dimensional *grid* in Cartesian coordinates.

A *real-world* environment is typically partially observable or unobservable, multiagent, dynamic, and continuous. Path planning in a partially observable or unobservable environment is difficult because the positions of obstacles are unknown to the agent. Nonetheless, in the real world the robot has no choice but to contend with these difficulties. Moveover, a real-world environment requires the use of physical robots which may be expensive, complex, and present many hardware challenges. As a result, autonomous robot systems are often tested in a *simulated* environment, as a simplified model on a computer, where a system can be tested many times under

identical conditions. A simulated environment allows researchers to select the characteristics of the environment to limit the complexity of the problem that their robot faces.

While most applications use robots on the ground in indoor and outdoor environments, there is also recent research on navigation for airborne and underwater robots. Those environments present additional challenges: the robot must contend with wind streams or underwater currents and must operate in three-dimensional space. An *unmanned aerial vehicle* (UAV) is an aerial robot that may operate autonomously. Originally developed for military applications, UAV systems usually also consider other hostile agents and such threats as radar and missiles. Underwater environments are typically unmapped and sensors are limited.

System design for autonomous robot navigation must make assumptions about the robot's environment: real-world or simulated, fully observable, partially observable, or unobservable, single agent or multiagent, static or dynamic, continuous or discrete, and on the ground, airborne or underwater. These choices determine the complexity of the environment and the difficulty of the path planning problem. This again makes comparisons among results developed for different types of environments difficult. Approaches discussed in this paper are for simulated grid environments with exceptions appropriately noted. The next section introduces heuristics and metaheuristics as ways to solve difficult problems like path planning.

1.3 Heuristics and Metaheuristics

A *heuristic* is an efficient strategy that can often solve a problem. Heuristic techniques may find satisfactory solutions when naive and brute-force approaches are too slow or fail to find an optimal solution (Pearl, 1984). In autonomous robot navigation, it is computationally expensive to find the shortest path to the target. Heuristics, such

as Euclidean distance and Manhattan distance, are used to estimate the distance to the target instead. Heuristics, however, are typically specific to a problem domain or problem class.

A *metaheuristic* is a broadly applicable technique that uses a heuristic strategy to obtain satisfactory solutions (Glover and Kochenberger, 2003). Examples of metaheuristics include ant colony optimization, genetic and evolutionary algorithms, simulated annealing, and tabu search. Surveys have reviewed the use of metaheuristics for optimization problems (Bianchi et al., 2009; Boussaïd et al., 2013). Metaheuristics are typically used when only incomplete or imperfect information is available, there is limited computational resources, or the problem is NP-hard. Metaheuristics are not problem-domain specific; they seek satisfactory solutions through efficient search. Many metaheuristics are tailored to avoid premature convergence.

A *hybrid metaheuristic* combines multiple heuristics and/or metaheuristics with other approaches to optimization problems to produce a better overall solution (Blum et al., 2011). These other approaches include dynamic programming, machine learning, constraint programming, tree search, and problem relaxation. Both metaheuristics and hybrid metaheuristics have been used to seek or improve solutions for path planning.

The remainder of this paper is organized as follows. Chapter 2 provides background on metaheuristics and reviews previous approaches in the use of single-solution metaheurstics for path planning. Chapter 3 describes population-based metaheuristics and hybrid metaheuristics and approaches for their use in path planning. Chapter 4 examines previous research in cognitive science on human and autonomous robot navigation. Chapter 5 concludes the paper and foreshadows future work.

2. Single-solution Metaheuristics for Path Planning

Metaheuristics have been used in robotics for path finding, navigation, and behavior optimization (Blum and Roli, 2003). This paper reviews the breadth of metaheuristics that have been used for autonomous robot navigation, whereas previous surveys only examined the use of specific metaheuristics for autonomous robot navigation (Manikas et al., 2007; Meyer et al., 1998; Bongard, 2013) or examined the use of metaheuristics more broadly across all of robotics (Fong et al., 2015).

Metaheuristic algorithms can be classified along two dimensions: the locality of the search and the number of alternatives considered at each iteration (Talbi, 2009). *Local search* explores the search space in the vicinity of an alternative, while *global search* has the ability to explore the entire search space. In search for an optimal solution, local search may become trapped in local optima, while global search may be impractical or impossible. Although both methods can find satisfactory solutions, local search may use fewer computational resources.

Throughout this paper, a *candidate* is a path found during search. A candidate's *neighbor* has a single action changed in, added to, or removed from the candidate's sequence of actions, with appropriate changes to the neighbor's states. A *single-solution* method maintains and improves one candidate at a time as it explores the search space. A *population-based* method maintains and improves a set of candidates as it explores the search space. This distinction between single-solution and population-based methods is often used to categorize metaheuristic algorithms (Blum and Roli, 2003).

The trade-off between *exploration* and *exploitation* is a fundamental challenge for search in AI problems. Algorithms balance exploration of unseen portions of the search space against exploitation of the knowledge collected in the search thus far to

focus the search near the current best candidate(s). Many methods and metaheuristics seek to optimize this trade-off to speed search.

The remainder of this chapter begins with background on basic search methods that are the building blocks of many metaheurstics. Pseudocode for these search methods was adapted from Luke (2013). The subsequent section discusses single-solution metaheuristics in the context of autonomous robot navigation.

2.1 Background

Most metaheuristics for robotics use some aspects of random search and hill climbing in their basic mechanisms (Fong et al., 2015). The consideration of multiple candidate solutions and the use of random restarts are also common. These four methods are discussed below.

Random search is a single-solution, global search method that repeatedly selects and examines a random candidate and stores a single best candidate as it searches. The benefit of random search is that it eventually explores the entire search space and so it avoids local optima. It may not, however, find an optimal solution in a reasonable amount of time.

Algorithm 1: Random search

```
Best ← an initial random candidate
repeat until Best is a satisfactory solution or computation time is exhausted
    S ← a random candidate
    if Quality(S) > Quality(Best) then
        Best ← S
return Best
```

Pseudocode for random search appears in Algorithm 1. Random search begins with an initial random candidate that it designates as the best candidate *Best*. It then repeatedly selects another random candidate. The *Quality* function computes

the candidate's quality based on domain-specific criteria. If the randomly selected candidate has a higher *Quality* value than *Best*, the new candidate becomes *Best*. This continues until *Best* is a satisfactory solution or the allotted computation time is exhausted. The final *Best* is returned. After an exhaustive search, *Best* is an optimal solution; otherwise random search may provide a satisfactory solution.

Random search can be used to find a solution for path planning. In an infinitely large search space, however, it is not possible to guarantee an optimal solution; a satisfactory solution must do. Random search does not use any strategies as it searches and is thus a naive method when used alone.

Hill climbing is a single-solution, local search method that seeks to continually improve a candidate. Whereas random search focuses only on exploration, hill climbing focuses only on exploitation. Algorithm 2 provides the pseudocode for hill climbing. Rather than select a new candidate at random, it selects a new candidate S that is an extension of the current best candidate *Best*. The *Extend* function randomly selects a neighbor of *Best*. Then the *Quality* function is used to compare S and *Best*, and the candidate with higher quality is retained. This continues until *Best* is a satisfactory solution or the allotted computation time is exhausted. Hill climbing continually extends a single candidate, whereas random search explores candidates throughout the search space.

Algorithm 2: Hill-climbing search

```
Best ← an initial random candidate
repeat until Best is a satisfactory solution or computation time is exhausted
    S ← Extend(Best)
    if Quality(S) > Quality(Best) then
        Best ← S
return Best
```

Hill climbing can also be used to find a satisfactory solution in path planning. Hill climbing's heuristic always goes towards a solution with higher quality. This strategy is naive because it can easily become trapped in local optima. For problems with a smooth convex search space, hill climbing will always eventually find an optimal solution, however, such a space is rarely the case. Two methods address this issue: steepest ascent and random restart. Both seek to strike a better balance between exploitation and exploration.

Steepest ascent hill climbing modifies hill climbing so that it considers multiple candidates and selects the one candidate with the largest improvement in quality. Pseudocode for steepest ascent hill climbing appears in Algorithm 3. Instead of one extension of *Best*, it considers n random extensions of *Best* at each iteration and keeps the candidate of greatest quality among the n extensions and *Best*. Although this method is still largely focused on exploitation, it explores more of the search space around the candidate.

Algorithm 3: Steepest ascent hill-climbing search

```
n ← number of extensions to consider on each iteration
Best ← an initial random candidate
repeat until Best is a satisfactory solution or computation time is exhausted
    S ← Best
    for n times do
        R ← Extend(Best)
        if Quality(R) > Quality(S) then
            S ← R
    end
    if Quality(S) > Quality(Best) then
        Best ← S
return Best
```

Hill climbing with random restarts modifies hill climbing so that it becomes a global-search method. Pseudocode for hill climbing with random restarts appears in Algorithm 4. In this method, hill climbing begins from a random candidate for a

randomly selected amount of time t. When that time is up, a new random candidate and a new amount of time are selected and hill climbing resumes. The candidate with the highest quality is retained. The process repeats until either a satisfactory solution is found or computation time is exhausted. Hill climbing with random restarts takes advantage of hill climbing while it also provides a way to escape from local optima. This search method balances exploration and exploitation more so than the previous methods but naively, since the restarts begin at random.

Algorithm 4: Hill-climbing search with random restarts

```
T ← distribution of possible time intervals
S ← an initial random candidate
Best ← S
repeat until
 Best is a satisfactory solution or overall computation time is exhausted
    t ← random time chosen from T
    repeat until Computation time t is exhausted
        R ← Extend(S)
        if Quality(R) > Quality(S) then
            S ← R
    if Quality(S) > Quality(Best) then
        Best ← S
    S ← a random candidate
return Best
```

Although random search, hill climbing, steepest ascent, and random restarts have all inspired and influenced metaheuristic techniques, they often suffer from premature convergence. The next section reviews single-solution metaheuristics and their use in autonomous robot navigation. These metaheuristics use a heuristic strategy to balance exploration and exploitation in a more deliberate and thoughtful way.

2.2 Simulated Annealing and Tabu Search

Simulated annealing is a single-solution, local-search metaheuristic (Kirkpatrick et al., 1983; Černỳ, 1985). It was inspired by the annealing process in metallurgy in which a heated material is slowly cooled. Simulated annealing seeks to balance the trade off between exploration of new areas of the search space and exploitation of the best candidate found thus far. It employs a mechanism similar to hill climbing but allows for the occasional selection of candidates with lower quality. To do so, a *temperature* controls the degree to which the lower quality candidates are selected. In this way, the search is biased towards exploration early in search, and then slowly shifts its bias towards exploitation as search progresses.

Algorithm 5: Simulated annealing

$t \leftarrow$ initial temperature value
$Best \leftarrow$ an initial random candidate
repeat until
Best is a satisfactory solution or overall computation time is exhausted or $t \leq 0$
  $S \leftarrow Extend(Best)$
  $a \leftarrow$ a random number in $[0, 1]$
  if $Quality(S) > Quality(Best)$ *or* $a < e^{\frac{Quality(S)-Quality(Best)}{t}}$ **then**
    $Best \leftarrow S$
  Update t
return *Best*

Pseudocode for simulated annealing appears in Algorithm 5. At each iteration, if the *Quality* value of the extended candidate S is greater than the *Quality* value of the candidate $Best$ then S always replaces $Best$. Otherwise, a temperature variable t governs the probability $e^{\frac{Quality(S)-Quality(Best)}{t}}$ that S replaces $Best$ even though S has a lower *Quality* value. This probability is always between 0 and 1 because it is used only when $Quality(S) \leq Quality(Best)$. Because the value of t decreases as the

algorithm explores the search space, the probability that a lower quality candidate is selected decreases over time.

Tabu search is a single-solution, local-search metaheuristic that also builds upon hill climbing and allows for the selection of lower quality candidates at each iteration (Glover, 1989, 1990). Unlike simulated annealing, however, tabu search temporarily forbids a return to recently visited candidates. Tabu search intentionally limits exploitation in an effort to increase exploration. Algorithm 6 shows the pseudocode for tabu search (adapted from Glover and Kochenberger (2003)).

Tabu search maintains an intially empty tabu list $Tabu$ of temporarily forbidden candidates. The algorithm begins with $Best$ and S, an initial random candidate. On each iteration, tabu search considers up to n neighbors of S. Among the neighbors that are not in $Tabu$, the candidate with the highest quality is saved as $bestNeighbor$. If all the neighbors of S are in $Tabu$ then candidates are removed from $Tabu$ until a neighbor of S is no longer in $Tabu$. (A method such as this that allows search to return to a previously visited candidate in $Tabu$ is called an *aspiration criterion*.) Next, if the quality of $bestNeighbor$ is greater than the quality of $Best$, $Best$ is replaced with $bestNeighbor$, and $bestNeighbor$ is inserted into $Tabu$ to prevent an immediate return to it. If the size of $Tabu$ is greater than a user-specified maximum tabu list length l then candidates are removed from $Tabu$ until the size is less than l. Typically, candidates are removed on a first in, first out basis. This allows a once-forbidden candidate to be reconsidered later. Finally, S is replaced with $bestNeighbor$ so that, on the next iteration, the neighbors of $bestNeighbor$ are considered. This process is repeated until $Best$ is a satisfactory solution or computation time is exhausted.

Tabu search can be modified to include a tabu tenure factor that allows candidates to be removed from the $Tabu$ list after a certain amount of time has elapsed. The tabu tenure is usually chosen randomly. *Diversification*, a focus on a portion of the

search space that shows promise, and *intensification*, a focus on previously unexplored areas of the search space, are other techniques that have been used in tabu search to improve performance.

Algorithm 6: Tabu search

```
l ← desired maximum tabu list length
Tabu ← [ ] the tabu list
Best ← an initial random candidate
S ← Best
n ← number of neighbors to consider at each iteration
repeat until Best is a satisfactory solution or overall computation time is
 exhausted
    Neighbors ← [ ]
    C ← [ ]
    Neighbors ←
     set of distinct candidates produced by n executions of Extend(S)
    C ← Neighbors \ Tabu
    if C ≠ ∅ then
        bestNeighbor ← candidate ∈ C with maximal Quality
    else
        // Else if all neighbors of S are in the Tabu list, remove candidates
         from Tabu until a neighbor is available, allowing the algorithm to
         return to a previously visited candidate
        repeat until bestNeighbor has a candidate
            Tabu ← Tabu \ {c} where c ∈ Tabu
            if c ∈ Neighbors then
                bestNeighbor ← c
    end
    if Quality(bestNeighbor) > Quality(Best) then
        Best ← bestNeighbor
        Insert bestNeighbor into Tabu
    if |Tabu| > l then
        repeat until |Tabu| < l
            Tabu ← Tabu \ {c} where c ∈ Tabu
    S ← bestNeighbor
return Best
```

Both simulated annealing and tabu search have been applied to path planning for autonomous robots in simulated 2D static environments. One approach for path

planning with simulated annealing compared three path representations (linear, a Bezier curve, and a spline interpolated curve) and found that all three were able to produce satisfactory paths if the cooling schedule for the temperature was tuned correctly (Tavares et al., 2011). Simulated annealing for path planning with circular obstacles has found satisfactory solutions and scaled well with the number of obstacles (Hayat and Kausar, 2015). Simulated annealing with multiple goals calculated satisfactory solutions for two UAVs in suitable time (Behnck et al., 2015). Finally, work with an online tabu search-based motion planner (Masehian and Amin-Naseri, 2008) and sampling-based collision detection with tabu search has done sensor-based path planning (Khaksar et al., 2012). Both approaches incorporated aspiration and diversification. Results showed satisfactory paths in convex, concave, and maze environments.

This section has reviewed two single-solution metaheuristics that have been used for robotic path planning: simulated annealing and tabu search. Both methods use heuristics to control the trade-off between exploration and exploitation to improve upon the performance of hill climbing. Although several approaches used these metaheuristics for path planning, the approaches have been evaluated in simplistic, unrealistic environments and no one approach has been thoroughly compared with other path planning approaches. The next chapter reviews population-based metaheuristics inspired by evolution and swarm behavior. A large body of research has applied those algorithms to path planning because they appear to find better solutions more efficiently.

3. Population-based Metaheuristics for Path Planning

Many population-based metaheuristics have been inspired by biological mechanisms and the behavior of organisms (Manikas et al., 2007). One broad category of population-based metaheuristics is *evolutionary algorithms*, which use mechanisms inspired by Darwinian principles, such as survival of the fittest and natural selection, to guide search. Recall that a candidate is a path found during search and a candidate's neighbor has a single action changed in, added to or removed from the candidate's sequence of actions, with appropriate changes to the neighbor's states. The first section reviews four kinds of evolutionary algorithms in the context of autonomous robot navigation: genetic algorithms, genetic programming, evolutionary programming, and differential evolution. Later sections discuss swarm algorithms and hybrid metaheuristics.

3.1 Evolutionary Algorithms

Inspired by biological inheritance, *genetic algorithms* are evolutionary algorithms that simulate mechanisms such as reproduction, mutation, recombination, and selection. A genetic algorithm creates a set of candidates that represent individuals in a population P. In each iteration, the population is called a *generation*. Genetic operators (e.g., reproduction, crossover, and mutation) are applied to the current generation to produce the next generation. A $fitness$ function is used to evaluate the quality of each candidate based on domain-specific criteria.

Pseudocode for a basic genetic algorithm appears in Algorithm 7, adapted from Back (1996). Initially, *Best* is the candidate with the greatest $fitness$ value in the initial population P. Then, on each iteration, two parents, p_1 and p_2 are randomly selected with replacement from the population, and p_1 and p_2 reproduce, via the *crossover* function, to produce two children, c_1 and c_2. (For *crossover*, both parents are par-

Algorithm 7: Genetic Algorithm

$s \leftarrow$ population size
$P \leftarrow$ a population of size s of randomly generated candidates
$P' \leftarrow [\,]$
$m \leftarrow$ mutation probability
$Best \leftarrow$ candidate with max $fitness$ in P
repeat until *Best is a satisfactory solution, there has been no significant change in P for many iterations, or overall computation time is exhausted*
  while $|P'| < s$ **do**
    $p_1 \leftarrow$ parent 1, a randomly selected candidate from P
    $p_2 \leftarrow$ parent 2, a randomly selected candidate from P
    $\{c_1, c_2\} \leftarrow crossover(p_1, p_2)$
    $c_1 \leftarrow mutate(c_1, m)$
    $c_2 \leftarrow mutate(c_2, m)$
    $P' \leftarrow P' \cup \{c_1, c_2\}$
  end
  if $fitness(Best) > \{\max_{c \in P'} fitness(c)\}$ **then**
    $P' \setminus \{\arg\min_{c \in P'} fitness(c)\}$
    $P' \leftarrow P' \cup \{Best\}$
  else
    $Best \leftarrow \{\arg\max_{c \in P'} fitness(c)\}$
  end
  $P \leftarrow P'$
  $P' \leftarrow [\,]$
return *Best*

titioned into two sections at the same split point. Then, c_1 is created from the first section of p_1 followed by the second section of p_2, and c_2 is created from the first section of p_2 followed by the second section of p_1.) After c_1 and c_2 are produced, they are subjected to the *mutate* function, which randomly changes each of them with probability m. They are then added to a new population P', the next generation. This process repeats until the size of P' reaches the population size s. Next, if the new population P' does not contain a candidate with higher $fitness$ value than $Best$, then the candidate with the lowest $fitness$ value in P' is replaced with $Best$. Otherwise, if the child population P' does contain a candidate with higher $fitness$ value than $Best$, then that candidate replaces $Best$. Finally, the population P is replaced with P' and the entire process is repeated. This continues until either $Best$ is a satisfactory solution, there is no significant change in the fitness of the candidates in P for many iterations, or overall computation time is exhausted.

Candidates in genetic algorithms are typically encoded as fixed-length strings. Each action is assigned a unique bit pattern, and the sequence of actions in a candidate is concatenated into a bit string. For example in path planning, if *Move* is encoded as 00, *Turn Left* as 01, and *Turn Right* as 11, then the candidate string 00 01 01 00 00 11 represents the sequence ⟨*Move, Turn Left, Turn Left, Move, Move, Turn Right*⟩. Genetic algorithms use fixed-length strings for candidates because it simplifies the data structures for P and P', and allows for straightforward crossover and mutation functions. Candidates in path planning rarely have the same number of actions, however, so a variety of different candidate encodings have been devised.

Instead of the uniform random selection of parents from P, many genetic algorithms use *roulette wheel selection*, which selects from P with probability proportional to the candidates' $fitness$ value. Another common method is *tournament selection* where, on each iteration, two candidates are randomly selected and the one with the

higher $fitness$ value is the one allowed to reproduce. Some genetic algorithms modify the *crossover* function so that it makes multiple splits instead of a single split, which complicates recombination.

Genetic algorithms have been used for path planning for autonomous robots. One approach compared simulated annealing, tabu search, and a genetic algorithm to minimize travel distance in a simulated, static, well-known environment (Hussein et al., 2012). Empirical evaluation showed that simulated annealing generated the shortest path fastest, the genetic algorithm found satisfactory solutions fastest, and tabu search was slowest. Other work developed tabu search for global path planning in small, medium, and large-scale grid environments with obstacles and compared it with a genetic algorithm (Châari et al., 2014). Experiments showed that tabu search found satisfactory solutions faster than the genetic algorithm in nearly all cases.

Another approach developed a multi-objective path planning method with a genetic algorithm that optimizes path length, path safety, and path smoothness in a simulated environment with static obstacles (Ahmed and Deb, 2013). This approach encoded a candidate as a variable-length integer vector, which represented the sequence of movements a robot makes in a 2D grid environment. Empirical results showed that the approach outperformed a single-objective genetic algorithm in environments up to 90% obstructed by obstacles. Another approach encoded candidates with a variable-length bit string for path planning in a simulated environment with static and dynamic obstacles; it produced shorter paths than a model with fixed-length encoding (Tu and Yang, 2003).

Other approaches have integrated domain-specific knowledge with genetic algorithm-based path planning. One method incorporated a grid representation of the environment and local search-based genetic operators for complex static and dynamic environments (Hu and Yang, 2004). Simulation results showed the incorporation of these

specialized genetic operators produced shorter paths more quickly than a genetic algorithm without them. Other work modified the mutation operator to prevent paths through obstacles and thereby avoid premature convergence (Tuncer and Yildirim, 2012). Experiments in simulation compared this mutation operator with several other mutation operators; results showed that this path-sensitive approach found more satisfactory solutions with better average fitness values in approximately the same time as the other methods.

Research has also explored the use of genetic algorithms to do path planning for UAVs in 3D environments. One approach represented the path as B-spline curves with the candidates for the genetic algorithm as the control points of the curves (Mittal and Deb, 2007). The approach considered multiple objectives, and in simulation it produced feasible paths under the requirement that the path pass through a specified point. Another method accounted for the physical and computational capabilities of the UAVs and mission specific objectives, and found optimal paths in 3D environments with multiple threats, again in simulation (Sanders and Ray, 2007).

Genetic algorithms have also been applied to path planning for cooperative multi-robot systems (Cai and Peng, 2002). This work introduces and adaptively updates a probability for crossover, evolves the candidate paths of individual robots in their own sub-populations, evaluates candidate fitness over the entire population, and encodes each candidate as a fixed-length decimal. Simulation results showed that the approach produced satisfactory paths for two robots. Another method for coordinated path planning with multiple UAVs accounted for the physical and computational capabilities of the UAVs and mission specific objectives (Zheng et al., 2005). In simulation it produced feasible routes for up to three UAVs in a 3D environment with multiple threats. A similar approach for coordinated path planning for multiple UAVs incorporated specialized genetic operators and evolved separate populations for each

UAV's path (Besada-Portas et al., 2010). Simulated scenarios showed the approach produced feasible paths for up to six UAVs in dynamic 3D environments with multiple threats.

While the approaches discussed above made simplifying assumptions and were tested in simulation, other work has considered hardware and implementation issues on physical robots. A genetic algorithm was implemented on-board a small robot with limited hardware resources in a static environment with obstacles (Burchardt and Salomon, 2006). The robot ran the algorithm in real-time and reached its goal in a simple environment with four obstacles. A parallel genetic algorithm on an embedded processor with a Field Programmable Gate Array (FPGA) produced smooth global path plans, was faster, and used fewer resources than it did without FPGA in an environment with static obstacles (Tsai et al., 2011).

Genetic programming is an evolutionary approach similar to genetic algorithms, but the population is a set of computer programs that solve a given problem rather than a set of strings (Koza, 1992). The goal of genetic programming is to find the computer program that best solves a given problem. These algorithms are typically written in a functional programming language such as Lisp. Each program is represented as a unit, and the genetic programming algorithm recombines, mutates, and selects among them at each iteration (Poli et al., 2008). The structure of a genetic programming algorithm is identical to Algorithm 7, except that candidates are computer programs.

Early work proposed genetic programming with Lisp for path planning to multiple targets on a factory floor (Yamamoto, 1998). Each computer program candidate was a sequence of functions, such as *move forward*, *move backward*, *turn right*, and *turn left*. The fitness function evaluated each program based on the total distance travelled.

Simulations showed the approach produced a satisfactory path in an environment with four targets.

A later approach had unique initial and goal positions for multiple robots, and sought satisfactory paths that avoided collisions (Kala, 2012). A candidate was a sequence of integers interpretable through a grammar as a program. The algorithm tried to optimize the paths of the individual robots separately, while the fitness function considered collision avoidance. The approach was able to produce collision-free paths for up to five robots in a simulated maze environment.

Recently, a genetic program did multi-objective path planning for UAVs (Yang et al., 2016). Each function in the program candidate is stored as a binary tree of programming operators that can be decoded into the direction of the next step in the path. A sequence of these functions represents a UAV's path, while the fitness function evaluates the path's length, flight altitude, and threat level. This method produced paths with better fitness values than a genetic algorithm in a simulated environment.

Evolutionary programming is similar to genetic programming, but addresses a single parameterized program. The search space is the set of programs created by the assignment of values to the program's parameters (Fogel, 1999). On each iteration, the population is a set of vectors, each of which represents an instantiation of the parameters. All the candidates in the population are mutated and the new population is selected from the parents and children based on their fitness values. Evolutionary programming repeats these steps until it finds a satisfactory candidate that optimizes the output of the program.

Differential evolution algorithms combine aspects of genetic algorithms and evolutionary programming (Storn and Price, 1997). Like evolutionary programming, the population of interest is sets of real-valued vectors, but these vectors are parame-

ters for a function rather than a program. On each iteration, the vectors undergo transformations similar to those in genetic algorithms. The major difference is that the recombination function creates a child as the sum of the weighted difference between two randomly selected population members and a third random population member (Price et al., 2006).

Differential evolution has also been used for path planning in autonomous robot navigation. One approach used differential evolution for coordinated navigation of UAVs in a static environment over the ocean (Brintaki and Nikolos, 2005). As in other multi-robot approaches, it assigned different initial positions to each UAV and produced a path plan for each UAV to a single target position that considered collision avoidance, multiple constraints, and multiple objectives. Simulation results showed that the approach produced feasible paths for three UAVs in a reasonable amount of time. A more recent study sought to optimize the tuning parameters of a differential evolution-based path planner for UAVs (Kok and Rajendran, 2016). It examined four tuning parameters (population size, differential weight, crossover rate, and number of generations) and found an optimal setting for these parameters that depended on a user-specified trade-off between minimal path cost and minimal computational cost.

This section has reviewed four evolution-inspired metaheuristics that have been used for robotic path planning: genetic algorithms, genetic programming, evolutionary programming, and differential evolution. These population-based metaheuristics all use Darwinian principles as heuristics to seek satisfactory solutions, but they differ their representation of the problem. The methods incorporate hill-climbing through selection, and randomization through mutation. The disadvantages of evolutionary algorithms are that there is no guarantee that the optimal solution will be found in finite time, that parameters like population size must be tuned by hand, and that maintenance of a large candidate population can be computationally and memory in-

tensive. The next section reviews population-based metaheuristics inspired by swarm intelligence that explore the search space with different heuristics.

3.2 Swarm Algorithms

Swarm intelligence metaheuristics are inspired by the crowd behavior of organisms such as ants and bees or the movement of particles (Bonabeau et al., 1999). These methods emulate highly motivated animals' strategies in their search for food. Swarm algorithms also incorporate mechanisms that allow candidates to communicate information to one another.

Ant colony optimization is a swarm-based metaheuristic inspired by the foraging behavior of ants (Dorigo et al., 2006). It simulates ants' ability to communicate indirectly through pheromones that mark the shortest route between their nest and food sources. Ants select a route to follow probabilistically based on the strength of the pheromones on the route. When an ant returns from a food source, it deposits pheromones in proportion to the amount of food it collected from that source.

Ant colony optimization represents the search space as a *state-space graph*, where a node is a state and an edge is an action that represents a transition from one state to another state. A solution begins at the node labeled as the initial state and traverses the graph to a node labeled as the goal state. A candidate for ant colony optimization is a finite ordered sequence of interleaved nodes and edges in the state-space graph. In this paper, only ant colony optimization uses this alternate definition of candidate.

Pseudocode for an ant colony optimization algorithm appears in Algorithm 8, adapted from Blum (2005). After the algorithm's parameters (the number of ants and pheromone trails) are initialized, on each iteration, the algorithm cycles through four phases. First, artificial ants are probabilistically assigned to candidates based on the pheromone values in the $ConstructAntSolutions$ function. Then the $ApplyLocalSearch$

Algorithm 8: Ant colony optimization

```
Set number of ants
Initialize pheromone trails
while Termination conditions not met do
    ConstructAntSolutions()
    ApplyLocalSearch()
    PheromoneUpdate()
    DaemonActions()
end
```

function improves each candidate through a local search at the end of the candidate. The *PheromoneUpdate* function then increases the pheromone values along the edges associated with promising candidates and decreases those associated with poor candidates. Finally, the *DaemonActions* function collects global information that can be used to bias the search process from a non-local perspective, such as an increase in pheromones for the best candidate found so far. These steps repeat until the ants find a satisfactory solution or overall computation time is exhausted. In path planning, the robot follows the satisfactory solution produced after ant colony optimization terminates. Each ant could be a robot exploring the physical space, however, a plan, by definition, is a sequence of actions selected prior to execution and so a robot exploring the space to find the goal is not path planning.

Ant colony optimization has improved a sub-optimal collision-free path in a real-time path planning algorithm (Guan-Zheng et al., 2007). Compared in simulation to a genetic algorithm, the ant-colony algorithm converged to the optimal path in fewer iterations and less time. Another approach for a static environment has also outperformed a genetic algorithm (Buniyamin et al., 2011). On average over five runs, the ant colony found the optimal path in fewer iterations and less time. Other work modified ant colony optimization for path planning with two kinds of ants: one kind explored the space randomly and left pheromones on promising candidates, while the

other explored only the candidates with high pheromone values (Yao et al., 2015). This method adjusted the number of the two types of ants to prevent premature convergence. In simulation this method outperformed two other versions of ant colony optimization and Dijkstra's algorithm.

An approach for underwater vehicles in a 3D environment had ants explore the state-space graph and keep track of their pheromone values for candidates independently (Yang et al., 2015). After all the ants found paths to the goal, the global pheromone values were updated. This modification sought to prevent the ants from herding toward potential local optima. Although it produced shorter paths than an unmodified ant colony optimization in simulation, search was not consistently faster.

An approach for path planning for cooperative multirobot systems with a genetic algorithm was improved in several ways (Qu et al., 2013). The fitness function considered path length, safety, and smoothness; several populations evolved in parallel, and it incorporated a new genetic operator and collision avoidance among robots. In simulation, the approach produced shorter paths faster than ant colony optimization for a single robot and was able to produce near-optimal paths for three robots.

The *artificial bee colony* is another swarm-based metaheuristic, this one inspired by the foraging behavior of honey bees (Karaboga and Basturk, 2007). An artificial bee colony algorithm simulates a colony of three groups of bees: employed bees, onlooker bees, and scout bees. A candidate path represents a food source for the bee colony. The quality or fitness of a candidate is measured by its amount of nectar. The hive represents a place where the bees congregate and exchange information. An artificial bee colony combines hill-climbing local search (through the employed bees) with global search (through the onlooker bees), and random search (through the scout bees).

An employed bee travels between a candidate in the search space and a dance area in the hive. There is a one-to-one mapping between employed bees and candidates. An employed bee's dance is based on the amount of its candidate's nectar. An onlooker bee watches the dances of employed bees when they return to the hive, and chooses a candidate based on the dances. A scout bee randomly searches the area around the hive for candidates. An employed bee's candidate is exhausted or abandoned when it cannot be further improved. An abandoned candidate is replaced by a candidate found by a scout bee.

Algorithm 9: Artificial Bee Colony

```
{E, O, S} ← Partition(B) // B is the set of all bees
e_i ∈ E ← initial unique random candidate c_i ∈ C
lim ← Number of iterations until a candidate is abandoned
repeat until
 Best is a satisfactory solution or overall computation time is exhausted
    Employed bees E greedily select between c_i and neighbor(c_i) based on the
     nectar amount
    if neighbor(c_i) is selected then
        neighbor(c_i) replaces c_i in C
    Onlooker bees O watch the dance of employed bees and visit a c_j ∈ C
     probabilistically
    Onlooker bees O greedily select between c_j and neighbor(c_j) based on the
     nectar amount
    if an employed bee has not moved from c_i for lim iterations then
        Scout bee randomly searches near the hive to discover a new candidate
         n
        Employed bee abandons c_i and is reassigned to n
        n replaces c_i in C
    Best ←
     candidate with maximum nectar among employed and onlooker bees
return Best
```

Pseudocode for a general artificial bee colony algorithm appears in Algorithm 9, adapted from Karaboga and Basturk (2007). The initial population is a random set of candidates. On each iteration, each subset of bees performs its respective procedure.

Each employed bee in E determines the amount of nectar in its candidate and in one neighbor of its candidate. If the neighbor has more nectar than its assigned candidate, then the bee reassigns itself to the neighbor.

Next, the employed bees return to the hive and dance to share their nectar information with the onlooker bees O. The onlooker bees assess this data from the employed bees, and each selects a candidate probabilistically. The more nectar in a candidate, the more likely an onlooker bee is to select it. An onlooker bee then travels to its selected candidate and evaluates the amount of nectar at a neighbor of that candidate. If the neighbor has more nectar than its selected candidate, then the onlooker bee selects it. Onlooker bees are assigned to candidates for a single iteration since they select new candidates probabilistically again in the next iteration. The candidate with the highest nectar amount among employed and onlooker bees is retained as $Best$. Thus, an onlooker bee's candidate is only retained if it becomes $Best$.

If any employee bee has not changed its candidate for lim iterations it abandons the candidate. The scout bees S randomly search for new candidates. If any employed bee had been assigned to a now-abandoned candidate then it is randomly reassigned to a new candidate found by a scout bee. The algorithm terminates either when $Best$ is a satisfactory solution or computation time is exhausted.

An approach used an artificial bee colony to find safe and smooth paths for multiple robots in an environment with static obstacles (Dou et al., 2014). The fitness function considered both path length and the distance between robots. Simulations with three robots found satisfactory paths faster than a standard artificial bee colony.

Particle swarm optimization (PSO) is inspired by the movement of flocks of bird and schools of fish (Kennedy and Eberhart, 1995). A PSO algorithm is similar to a genetic algorithm, but it does not use evolutionary functions, such as crossover and

mutation. Instead, it maintains a population of candidates, called *particles*, that move through the search space to seek a satisfactory solution. A fitness function evaluates the quality of the particles. Each particle performs a local search whose direction is influenced by the other particles.

A PSO is analogous to bird flocking, where each particle is a bird in the search space. The birds search for a hidden food source, which represents a satisfactory solution with a minimally acceptable fitness value. They do not know where the food source is, but each bird senses how close it is to the food as the difference between its current fitness value and the minimally acceptable fitness value. The bird closest to the food source squawks loudest, and all the other birds move toward it. If any other bird gets closer to the food source then that bird squawks still more loudly and the birds are drawn toward it instead. This continues until one of the birds finds the food source.

The pseudocode for a PSO algorithm appears in Algorithm 10, adapted from Kennedy and Poli and their colleagues (Kennedy et al., 2001; Poli et al., 2007). PSO is initialized with a group of random particles P. A particle's position is a path in the search space. Each particle $p_i \in P$ has a best known local position L_i, the position with the highest fitness value it has seen thus far in the search. The population's best known global position $GBest$ is the position with the highest fitness value across all the particles. A particle's movement through search space is determined by its velocity vector v_i, calculated as a weighted sum of L_i and $GBest$. On each iteration, the best known local positions L_i are updated, $GBest$ is updated, and then the particles move based on their velocity v_i. This repeats until $GBest$ is a satisfactory solution, there has been no significant change in the fitness values of P for many iterations, or overall computation time is exhausted.

In one PSO approach to robot navigation, a path was represented with Ferguson cubic splines and then PSO optimized the parameters of the splines to find a path around the obstacles (Saska et al., 2006). In simulation, the method produced feasible and shorter paths than two traditional nonmetaheuristic methods. One approach with an artificial bee colony constructed a path described by cubic Ferguson splines and then modified the parameters of the splines to find an optimal path in an environment with static obstacles (Mansury et al., 2013). In simulation, this approach converged to an optimal path faster than either a genetic algorithm or PSO. In another approach, path planning was formulated as a time-varying nonlinear programming problem with four dimensions and then a PSO was used to solve the reformulated problem (Ma et al., 2013). Simulations in an environment with static and dynamic obstacles showed that the approach quickly converged to a satisfactory solution.

Algorithm 10: Particle Swarm Optimization

$s \leftarrow$ population size
$P \leftarrow$ a population of size s of randomly generated particles p
$L \leftarrow$ the fitness value and position of the initial particles in P // Best local known position
$GBest \leftarrow \{\}$
repeat until *GBest is a satisfactory solution, there has been no significant change in the fitness values of P for many iterations, or overall computation time is exhausted*
  for *particles* $p_i, i = 1...s$ **do**
    if $fitness(p_i) > fitness(L_i)$ **then**
      $L_i \leftarrow \{fitness(p_i), position(p_i)\}$
  end
  $GBest \leftarrow position(\arg\max_{l \in L} fitness(l))$ // Position with max fitness in P
  for *particles* $p_i, i = 1...s$ **do**
    $v_i \leftarrow velocity(position(L_i), GBest)$ // Determine particle velocity with a weighted combination of local best position and global best position
    $updatePosition(p_i, v_i)$ // Update particle position using velocity
  end
return *GBest*

A multi-objective PSO algorithm for path planning considered terrain, danger sources, and path length (Geng et al., 2013). This algorithm partitioned the environment and combined paths from each partition. The approach outperformed a genetic algorithm in simulation and successfully guided a robot in a real-world outdoor environment with static obstacles and different terrains. A case study in simulation compared the performance of multi-objective path planning algorithms for UAVs with multiple performance metrics and graphical representations and found that differential evolution outperformed planners based on genetic algorithms and particle swarms (Besada-Portas et al., 2013).

A PSO approach for a single UAV in a 3D environment replaced the Newtonian rules for particle movement with quantum mechanics and the velocity vector with a phase angle vector that maps the position of particles (Fu et al., 2012). In simulation, this method produced lower cost paths more quickly than a genetic algorithm, differential evolution, and a standard PSO. Another approach parallelized a real-time genetic algorithm for UAV path planning in a 3D environment on multicore CPUs (Roberge et al., 2013). The trajectories produced by this algorithm in simulation significantly outperformed a similar algorithm based on PSO.

An artificial bee colony for a UAV was modified to consider distances to threats (Li et al., 2013). It allowed the UAV to enter threat regions if the fitness of the path outweighed the risk. Simulation results showed that this approach converged to an optimal solution faster than PSO. Another approach with an artificial bee colony had an objective function that addressed the distance to the goal, obstacle avoidance, and robot collision avoidance (Liang and Lee, 2015). It modified the artificial bee colony with an elitist strategy, so that the best candidate influenced the direction of search and adaptively changed the population size. The approach produced paths for five robots in a simulated environment with static obstacles. It was also tested

in another problem domain where it outperformed a genetic algorithm and PSO in the search for a global optimum of twelve convex and nonconvex multidimensional numerical functions (e.g., find the value of x that minimizes $f(x) = \sum_{i=1}^{30} x_i^2$, where $-100 \leq x_i \leq 100$).

A recent approach implemented a differential evolution path planning algorithm for a UAV in a 3D environment with constraints (Zhang and Duan, 2015). For candidate selection, it evaluated each potential solution based on how well it satisfied the constraints. Over 50 test runs, this approach produced paths with the lowest average cost and always found a path that met all constraints, compared to a PSO, an artificial bee colony, and four other versions of differential evolution.

Other work has considered multi-robot path planning with differential evolution but from a distributed perspective (Chakraborty et al., 2009). Rather than determine paths for all robots simultaneously, it constructed paths for each individual robot separately, with a fitness function that considered path length and potential collisions. This approach was tested in simulation with up to 14 robots and produced better paths than a centralized-version of differential evolution and PSO. Another approach used an artificial bee colony for path planning for multiple robots. In an environment with static obstacles, a fitness function considered both the shortest paths for all robots and collision avoidance (Bhattacharjee et al., 2011). This approach outperformed differential evolution and PSO in a simulated environment with ten robots.

A *firefly algorithm* is inspired by the flashing behaviour of fireflies, where a firefly is attracted to other fireflies in proportion to the brightness of their flashes. One path planning approach for a UAV modified the firefly algorithm to allow fireflies to exchange information (Wang et al., 2012b). Over 100 simulations, the approach minimized the objective function most often on average and in approximately the

same amount of time in comparison with ant colony optimization, PSO, differential evolution, three different genetic algorithms, and two other evolutionary-based metaheuristics. A *bat algorithm* is inspired by the echolocation behaviour of bats, which vary the frequency, loudness and rate of pulse emissions to attract or repel other bats. A bat algorithm for a UAV was modified with a mutation operator and, in simulation, outperformed the same set of metaheuristic methods as the modified firefly algorithm approach (Wang et al., 2012a).

A *cuckoo search algorithm* is inspired by the parasitic behavior observed in some cuckoo species that lay eggs in the nests of other bird species. The algorithm represents a candidate as an egg, and these eggs are stored in host nests. A cuckoo attempts to replace an egg in some nest with its own egg (a new candidate). The host recognizes and removes the impostor egg with some probability. The best nests (those with high quality eggs) are carried over to the next generation. This continues until a satisfactory solution is found or overall computation time is exhausted. Cuckoo search in an unknown environment with static obstacles produced shorter paths than a genetic algorithm and PSO in simulation (Mohanty and Parhi, 2016). It also controlled a physical robot in real-time to avoid obstacles and reach its goal successfully in a static, partially observable, simple small indoor environment.

Finally, a few population-based metaheuristics are neither evolutionary nor swarm-based. A *gravitational search algorithm* is based on Newtonian physics. One example planned paths for a UAV with an improved memory mechanism and a modified weighting scheme (Li and Duan, 2012). It converged to a better solution more quickly than a genetic algorithm or a PSO in a simulated environment with static obstacles. *Harmony search*, inspired by the improvisation process of jazz musicians, has also been used for path planning in a static environment (Panov and Koceski, 2013). It

guided a 3-wheel robot in real-time, and in simulation produced feasible solutions faster than ant colony optimization or a genetic algorithm.

This section has reviewed swarm algorithms and other metaheuristics inspired by natural phenomenon and physical processes, and described approaches that use these methods for robotic path planning. Swarm algorithms incorporate communication mechanisms, such as pheromones, dances, and squawks, so that local information can influence the entire population. Evolutionary algorithms lack an equivalent mechanism. Swarm algorithms, however, still face the same issues as evolutionary algorithms: no guarantee of an optimal solution in finite time, performance that is highly dependent on parameter tuning, and computational intensity. As a result, researchers have sought to combine population-based metaheuristics with each other or with other methods. The next section reviews the use of such hybrid metaheuristics for path planning.

3.3 Hybrid Metaheuristics

A hybrid metaheuristic combines multiple metaheuristics or uses a metaheuristic along with another approach (e.g., machine learning). In an effort to resolve the shortcomings of individual metaheuristic methods, hybrid approaches have been used for path planning in autonomous robot navigation. Typically, the hybrid metaheuristic takes one of two forms: either each component is used separately for a different part of the navigation system or components of both methods are blended together to create a new, hybridized system. For example, a hybrid of the first type could use a genetic algorithm for path planning and a PSO for obstacle avoidance, whereas a hybrid of the second type could use genetic operators to modify the particles in a PSO for path planning. This section focuses first on combinations of two metaheuristics and then on those that combine a metaheuristic with other techniques.

Several hybrid metaheuristics combine two metaheuristics for a single robot. One method modified ant colony optimization to include a crossover operator from a genetic algorithm (Châari et al., 2012). Simulations in a static environment showed that the approach was faster and generated shorter paths than ant colony optimization and a genetic algorithm. In another approach, PSO was combined with *biogeography-based optimization* (*BBO*), an evolutionary metaheuristic inspired by the distribution of biological species through time and space, for path planning in a static environment (Mo and Xu, 2015). Experimental results in simulation showed that this hybrid method outperformed BBO, a genetic algorithm, and PSO.

A *memetic algorithm* is inspired by memes, ideas or behaviors that spread from person to person within a culture (Moscato and Cotta, 2003). Because they incorporate some aspects of genetic algorithms, memetic algorithms are sometimes considered hybrid metaheuristics. A memetic algorithm maintains a population of candidates and refines this population over multiple generations. A candidate is improved with local search techniques before it is added to the next generation. One approach combined a memetic algorithm with a bacterial evolutionary algorithm, inspired by microbial evolution (Botzheim et al., 2012). It replaced the genetic operators of mutation and crossover with bacterial mutation and gene transfer operators. The approach was able to direct two different kinds of robots in several partially observable, indoor, real-world environments with static obstacles. In simulation, it outperformed a genetic algorithm and an unmodified memetic algorithm.

Dual hybrid metaheuristics have also been applied to multi-robot path planning. A recent approach combined an improved PSO and improved gravitational search to update the position of particles with both the velocity from PSO and the acceleration from gravitational search (Das et al., 2016b). A similar approach used differential evolution to perturb particle velocities in PSO (Das et al., 2016a). Both approaches

converged to a better solution faster than a PSO, gravitational search, or tabu search in a simulated 2D environment with up to 25 robots. They also guided two physical robots through a partially observable, simple, small, indoor environment with static obstacles.

Hybrids of two metaheuristics have also planned paths for a UAV. In simulation, ant colony optimization was modified with differential evolution to optimize the *PheromoneUpdate* step (Duan et al., 2010). It produced shorter paths more quickly than unmodified ant colony optimization. Another approach updated the velocity in a quantum-based PSO with modified differential evolution for a UAV at constant altitude (Fu et al., 2013). This approach converged faster and produced lower cost paths on average over 20 simulations in comparison to a genetic algorithm, differential evolution, and PSO. A third method ran multiple PSO and genetic algorithms in parallel on a multicore CPU and kept the best candidate across the various populations (Roberge et al., 2014). In simulation, this approach produced significantly lower cost paths over 50 trials compared to a genetic algorithm and PSO.

Two approaches for a UAV used the genetic operators from differential evolution to modify the way cuckoo search selects a cuckoo (Wang et al., 2012c) and the way a bat algorithm selects a bat (Wang et al., 2016). The first approach outperformed ant colony optimization, differential evolution, three different genetic algorithms, PSO, BBO, and evolutionary strategy (another evolutionary algorithm) in a simulated 3D environment with five obstacles. The second approach converged to an optimal 3D path in simulation faster than an unimproved bat algorithm, and produced approximately the same result as the first approach in the same environment.

Table A1 (Appendix) summarizes the approaches that combine two metaheuristics discussed above. There is no clearly superior method, because the approaches tested different numbers and types of robots in different environments. The only pattern

is that all but two approaches combined an evolutionary algorithm with a swarm algorithm. As mentioned earlier, this may be due to an effort to draw on the unique strengths of each method. Since most of this work is recent, there is still significant room for creative new combinations and improvements on those approaches. Any new methods, however, should be evaluated in real-world environments with physical robots to show the feasibility of actual deployment.

Metaheuristics have been combined with other exact and heuristic-based methods to do path planning for autonomous robot navigation. *Rough set theory*, a mathematical tool to deal with imprecise, inconsistent, and uncertain information, has been used to improve the initial population for a genetic algorithm (Wu et al., 2006). Simulation experiments showed that the approach was faster than a genetic algorithm with binary encoding and converged to a solution with higher fitness in a simulated environment with up to 30 static obstacles. Genetic algorithms have also been adapted for path planning in unusual environments. Work for an autonomous underwater vehicle incorporated the space-time variability in an ocean environment through two novel genetic operators (Alvarez et al., 2004). The population of candidates was initialized with *dynamic programming*, a method that breaks a complicated problem down into simpler sub-problems recursively and stores and solves those sub-problems. This approach was able to produce an optimal 3D path in a simulated environment with dynamic currents. Another approach generated multiple populations of paths for a UAV with a genetic algorithm and then used linear programming to combine these paths (Arantes et al., 2016). In simulation, the approach quickly found robust solutions in 50 different 2D environments with non-convex static obstacles.

Other work for multiple robots combined a memetic algorithm with differential evolution to improve the global search coverage and *reinforcement learning*, a machine learning method in which an agent learns how to act through interaction with

the environment, to refine the candidate locally (Rakshit et al., 2013). This approach outperformed differential evolution, PSO, and a genetic algorithm in a partially observable, simple, small, real-world environment with two robots and five static obstacles. In another approach, PSO was used for path planning for a single robot with two objectives, path length and smoothness, and it was combined with the *probabilistic roadmap method*, a sampling-based method that builds a network graph of an environment, for obstacle avoidance (Masehian and Sedighizadeh, 2010b). Another approach combined a probabilistic roadmap with a negative PSO, a velocity method that seeks to avoid the worst particle position rather than move towards the best particle position (Masehian and Sedighizadeh, 2010a). The approaches were faster and produced shorter paths than the probabilistic roadmap method with Djikstra's algorithm and an unmodifed PSO.

A recent approach combined an artificial bee colony with evolutionary programming for path planning in an environment with static obstacles (Contreras-Cruz et al., 2015). The artificial bee colony component created a set of collision-free paths from the initial position to the goal position, and then the evolutionary programming component optimized these initial paths based on path length and smoothness. A candidate path was represented as a sequence of positions. The mutation operator had four ways to modify a path: delete a position, smooth the path around a position, move a position to a new collision-free position, or delete all the positions between two randomly selected positions. The approach produced shorter, smoother paths in less time than a probabilistic roadmap method in 46 simulated small and large environments with differing amounts of realistic and artificial obstacles, and was able to successfully guide a physical robot to its goal around two static obstacles in a small, simple indoor environment.

Chaotic search, based on nonlinear chaotic system dynamics, has been used to avoid local optima (Caponetto et al., 2003). As opposed to random search which explores the search space uniformly, chaotic search uses ergodicity and stochasticity to explore the search space nonlinearly without repetition. One approach for a UAV used chaotic search to replace local search in an artificial bee colony algorithm (Xu et al., 2010). It converged more quickly to a lower-cost path than a standard artificial bee colony in a simulated 2D environment with static and dynamic obstacles. Another approach for a 3D environment adapted PSO with a fitness-scaling method, adaptive parameter adjustment, and a chaotic search method (Zhang et al., 2013). Fitness scaling transforms fitness values with ranking and power functions. This approach converged towards the best path faster than a genetic algorithm, simulated annealing, or an artificial bee colony. A third method for a 3D environment used chaotic search for adaptive parameter adjustment in a real-time PSO (Cheng et al., 2014). Compared in simulation to a genetic algorithm, simulated annealing, an artificial bee colony, and a standard PSO, this approach had lower average computation time and a higher success rate.

Fuzzy logic reasons from incomplete, ambiguous, distorted, or inaccurate information with a range of logical values. One path planning method combined genetic operators based on domain knowledge and a fuzzy logic control strategy to adaptively adjust the mutation and crossover probability in a genetic algorithm (Li et al., 2006). In simulation of static and dynamic environments, this approach produced shorter paths in less time than another genetic algorithm. Another approach combined ant colony optimization with a fuzzy logic inference system for path planning among static and dynamic obstacles (Garcia et al., 2009). The ants accounted for the distance between the source and target nodes in the $PheromoneUpdate$ step, and

the paths were evaluated with a fuzzy cost function. This approach was faster than standard ant colony optimization in simulated experiments.

Artificial potential field (APF) is an obstacle-avoidance method that represents the environment as an artificial force field where the robot is attracted to the goal and repulsed by obstacles. One method combined a genetic algorithm with APF to ensure candidates were collision-free (Miao et al., 2011). Simulations showed that the approach performed better when the population of candidates was diverse. Another approach modified APF to plan for a 6-wheeled rover in a 3D rough terrain environment (Raja et al., 2015). In this approach, a genetic algorithm minimized a cost function that considered such rover constraints as roll, pitch, and yaw limitations, to optimize the weights of the potential field function. Experimental results with an actual rover on terrain with coarse sand and small rocks showed that the rover successfully followed feasible paths produced offline by the approach. A recent approach combined a modified PSO with APF and a fuzzy logic controller for an environment with static obstacles (Kuo et al., 2016). This approach outperformed a genetic algorithm, gravitational search, and PSO in simulation, and successfully guided a robot in a simple, real-world environment with static obstacles.

Inspired by biological nervous systems, an *artificial neural network* is a machine learning paradigm with a structure that consists of multiple connected layers of nodes. A challenge with artificial neural networks is how to determine the best structure to use. An early approach used a genetic algorithm to evolve the structure of a neural network (Cliff et al., 1993). In simulation, the neural network planned a path to the center of a cylindrical arena with no obstacles. A later approach modeled a collision penalty function with a neural network and then used it with an improved simulated annealing algorithm to do path planning (Gao and Tian, 2007). The approach pro-

duced shorter paths in less time than simulated annealing in a simulated environment with static obstacles.

Table A2 (Appendix) provides an overview of the approaches reviewed above that combine a metaheuristic with a non-metaheuristic method. These methods address different environments with different types of robots and most are evaluated in simulation. Generally, many different kinds of hybrid metaheuristics have been used for path planning, however, there is no single best combination. Each author presents an algorithm, validates its performance in a few test environments, and sometimes compares it with a few other methods. As a result, no single approach has risen above the rest. Moreover, no single type of hybrid metaheuristic has seen long-term sustained research. Instead, different types have had bursts of popularity. Thus, there is significant potential to develop new approaches. Once again, any new work should be evaluated in real-time applications for physical robots.

This chapter has reviewed evolutionary algorithms, swarm algorithms, hybrid metaheuristics, and their use in autonomous robot navigation. The next chapter considers cognitive approaches to autonomous robot navigation as an alternative to the metaheuristic-based approaches seen thus far. Cognitive approaches draw upon human behavior to improve the navigation abilities of a mobile robot, particularly in an environment where people are present.

4. Cognitive Models for Autonomous Robot Navigation

Cognitive science studies the mind and intelligence (Friedenberg and Silverman, 2011). It spans multiple disciplines, including philosophy, psychology, artificial intelligence, neuroscience, linguistics, and anthropology (Miller, 2003). Cognitive science includes the construction of computational models that seek to explain human behavior. These models can imitate techniques from human cognition to improve the behavior of artificial agents. *Spatial cognition* is the subfield of cognitive science that studies navigation and *wayfinding*, spatial problem solving employed by people to solve navigation problems (Wolbers and Hegarty, 2010).

This chapter first covers heuristics people use in navigation and how they decide what action to take. The next section reviews cognitive models of human navigation and wayfinding. It also discusses autonomous robot navigation systems inspired by human behavior.

4.1 Human Navigation Heuristics

This section reviews results from spatial cognition research on heuristics that people use in navigation and wayfinding. A *decision* is a choice among a set of alternatives. *Decision making* is the process by which an agent selects an alternative. *Reasoning* is the process of drawing a conclusion from information to solve a problem or make a decision (Leighton, 2004).

Heuristics are used by people to make fast and frugal decisions, especially when constrained by limited time, knowledge, and computational power (Gigerenzer et al., 1999). It has been argued that much of human reasoning and decision making can be modeled by these heuristics without complex deterministic or probabilistic models that account for all available information. In other words, people have *bounded*

rationality. Gigerenzer proposes three types of heuristics: those that guide search for alternatives, those that determine when to stop the search, and those that make a decision given the results of the search. Gigerenzer also suggests that people employ *cognitive economy*, that is, they tend to employ the heuristic with the least cognitive cost.

The spatial environments in which people navigate are complex and dynamic. People rarely have perfect information about the environment so they cannot make optimal navigation decisions. Nonetheless, they are still able to travel through challenging environments with good-enough decisions (Conlin, 2009). People use a variety of heuristics in navigation and wayfinding to search for paths, and make decisions when confronted with unknown environments and obstacles. Most heuristics for navigation are either triggered by external information in the environment or by an internal signal. These heuristics have been examined in outdoor, indoor, and virtual environments. The wayfinding heuristics that people use in indoor environments have been shown to be similar to and correlated with the heuristics they use in outdoor environments (Lawton, 1996). A *virtual environment* is a visual or audiovisual artificial representation of an environment that provides an opportunity to examine a person's behavior in simulation. In virtual environments that closely resemble the real world, people use wayfinding heuristics similar to those they used in the associated real-world environment (Darken and Sibert, 1996).

People adeptly choose the best strategies with respect to the problem they confront. Consider, for example, a person who travels to a hospital emergency room and a person who walks through a park to enjoy the scenery around a lake at the center of the park. Both are faced with a navigation problem, how to get from their current location to a target location, but each of them employs significantly different navigational strategies to solve that problem. It has been argued that human be-

havior is goal-directed (Aarts and Elliot, 2012) and that the navigation instructions people give depend upon the recipient's activity (Hirtle et al., 2011). Additionally, it has been shown that a person's wayfinding strategy when she travels through an environment differs from her strategy to plan a prospective path, or to give route directions to someone else (Hölscher et al., 2011).

Moreover, within a particular navigation problem, a person may apply different strategies at different points in the task. A study based on scans of brain activity provided evidence that people spontaneously shift between different strategies during a navigation task (Iaria et al., 2003). For example, the person in the park may, at one point, follow a trail at a leisurely pace and, at another point, leave the trail to avoid a fallen tree. While on the trail, her strategy could be to walk at a normal pace and stay in the center of the trail. When she leaves the trail to avoid the fallen tree, her strategy may change so that she moves more slowly and more cautiously through the brush. People can adapt their strategies to their overall goal, their current state, and their environment.

People also acquire spatial knowledge as they navigate. This information is used both to reason directly and to build internal representations from which to reason. A visual cue, such as a *landmark* (an interesting and distinctive object in the environment), is one type of spatial knowledge that can be used to localize and plan paths (Richter and Winter, 2014). Examples of visual cues include statues, signs, the location of the sun, shadows, and colors. People use visual cues to determine their position in the environment by triangulation or from bearing and distance. Without visual cues, localization becomes more difficult and people must rely on other methods to determine their position. Visual cues are also used in path planning as reference points in the environment. People follow a path more successfully if the waypoints are semantically meaningful landmarks rather than uninteresting or unremarkable places.

A *frame of reference* is a representation of objects in an environment in relation to some coordinate system. An *egocentric* frame of reference represents locations with respect to the perspective of an agent, whereas an *allocentric* frame of reference represents locations within a framework external to the agent, regardless of the agent's position (Klatzky, 1998). Examples of allocentric frames of reference include the geographic coordinate system of latitude and longitude, and the cardinal directions of north, south, east, and west.

Spatial orientation is the ability to maintain an internal representation of one's heading with respect to an allocentric frame of reference of the environment (Richter and Winter, 2014). *Path integration* is the process by which velocity and acceleration information are used to update the agent's position as it moves through an environment (Loomis et al., 1999). Path integration provides localization independent from any visual cues in the environment. As they move through an environment, people also acquire *route knowledge*, a sequence of locations and landmarks along a path, from an egocentric reference frame and *survey knowledge*, the spatial layout of the environment that includes relations between locations, from an allocentric reference frame (Latini-Corazzini et al., 2010).

A *cognitive map* is a mental representation of an environment a person builds as she moves through that environment (Golledge, 1999). Rather than try to remember and reason over all the sensed details of their environment, people use this compact and meaningful representation to reason and reduce their cognitive load. A cognitive map uses an allocentric representation of the environment that incorporates landmarks, route knowledge, and survey knowledge (Tversky, 1993). Landmarks represent points in the map, routes represent as lines that connect locations in the map, and survey knowledge determines the spatial relations in the map. It has been argued that cognitive maps use metric distances and angles (Gallistel, 1990). Later

work, however, argues that cognitive maps use a non-metric qualitative topological structure (Foo et al., 2005).

People also use external information, such as maps, photos, verbal descriptions, or route directions, as heuristics to form an internal representation of the environment prior to navigation. One study in an indoor environment found that individuals who studied a map learned a route faster than those who read verbal descriptions (Pazzaglia and De Beni, 2001). The use of external information, however, can increase cognitive load as individuals try to reconcile the external information with their perception of the environment. People may use a defensive wayfinding strategy if a mismatch occurs between the perceived environment and received route directions (Tomko and Richter, 2015). Additionally, people's experience and knowledge from other, similar environments contributes to their internal representation in a new environment. For example, a person who enters an unknown building does not start with a blank slate, but uses her previous experience of navigation in buildings and knowledge about building conventions (e.g., that rooms are arranged around corridors, and that elevators allow travel to different floors).

The remainder of this section describes heuristics and strategies that have been observed in human studies. Heuristics used to find and select a path plan are discussed first, followed by heuristics used in the course of movement through the environment. Finally, human characteristics and their impact on navigation strategies are outlined.

Before people can plan a path, they must select the criteria that will guide their search for a plan. It has been observed that people use many different criteria to choose paths, and that the criteria they select depends upon their motivation (Golledge, 1999). Table 1, adapted from Golledge, shows some of these path-selection criteria. One general strategy in path planning is to take the shortest path, but it is not clear how people measure path length. One study proposed three dis-

tance measures for outdoor movement through a city environment: metric distance, number of turns, and angular change (Hillier and Iida, 2005). Empirical examination of pedestrian movement found that paths minimized the number of turns or the angular change. Another study found that commuters to work tended to follow a path with shorter travel time rather than shorter distance (Zhu and Levinson, 2015). The criteria a person selects are heuristics that directly impact search for a plan.

Minimize path length	Minimize effort
Minimize travel time	Longest leg first
Minimize the number of turns	Shortest leg first
Maximize the number of turns	First noticed path
Minimize the number of curved segments	Novel path (different from previously travelled)
Maximize the number of curved segments	Avoid congestion
Minimize the number of segments in the chosen path	Avoid detours
Minimize the number of nonorthogonal intersections	Minimize negative externalities (e.g., pollution)
Minimize actual or percieved cost	Maximize aesthetics

Table 1: Path-selection Heuristics

Human spatial memory is organized hierarchically in nested levels of detail. This organization influences path planning and navigation behavior (Wiener and Mallot, 2003). In a study with human subjects in a virtual environment, environmental regions that represent this hierarchical organization were shown to be perceived and encoded very early in the process of learning an environment (Wiener et al., 2004). The study observed the use of three path planning strategies: the fine-to-coarse planning heuristic, the least-decision-load strategy, and the cluster strategy. It found that a linear combination with equal weights was able to predict the subjects' navigation behavior. The fine-to-coarse planning heuristic makes a detailed plan for close surroundings and an abstract plan (at the region level) for distant locations. The least-decision-load strategy plans a paths that minimizes complexity. The cluster strategy, when faced with multiple goal locations, prefers to visit spatially clustered targets first. It was argued that regional knowledge structures space and it also al-

lows the employment of search strategies to overcome missing or imprecise spatial knowledge. This conclusion was validated with participants in a physical replica of a virtual environment (Wiener et al., 2009).

Another study of wayfinding behavior compared participants' strategy preferences in navigation through real-world multi-level buildings (Hölscher et al., 2006). The study compared a *central point strategy* (reliance on paths through well-known parts of the building), a *direction strategy* (reliance on paths that go first towards the horizontal position of the goal), and a *floor strategy* (reliance on paths that go first towards the vertical position of the goal). Participants preferred the floor strategy; those that used it arrived at the goal with shorter average travel distance and times. In a follow-up study, these strategies were evaluated in a multi-level multi-building environment (Hölscher et al., 2009). In this setting, the participants reached their goal more effectively when they first followed the direction strategy, because it was more efficient to go to the correct building before the correct floor. This behavior seems to reflect the observation by Wiener et al. (2004) that people perform hierarchical planning across regions in the environment.

People also use spatial knowledge learned during navigation to inform future wayfinding tasks. A *shortcut* is a metrically shorter path to a location, compared to a previously known path to that same location. In an experiment with immersive virtual reality, participants found shortcuts in the environment only when landmarks were present (Foo et al., 2005). A follow-up study reaffirmed that people use landmarks, rather than metric survey knowledge, to find shortcuts (Foo et al., 2007).

Several path planning heuristics have been observed for people who travel to multiple locations. The nearest neighbor strategy is a heuristic that repeatedly chooses the closest target from the current location until all the targets have been visited (Gärling et al., 1988). When the goal is probabilistically located at one of several positions,

the rich-target-first strategy states has participants visit the positions in order of the likelihood that they contain the goal (Wiener et al., 2008). In another study, participants were shown to use their background knowledge about the spatial relationship between objects in a supermarket to plan a path to get to all the objects (Kalff and Strube, 2009).

Once a plan was selected, one study found two heuristics that were used to ensure correct traversal of the path: check if the perceived visual cues along the path conform with the expected visual cues in the plan, and plan multiple alternative paths in case a conflict arises (Spiers and Maguire, 2008). People also use heuristics when confronted with an unexpected obstacle during execution of the plan. Subjects were shown a path to a goal in a virtual maze and told to follow that path, but were then impeded by an obstacle (Janzen et al., 2001). To return to the learned path, subjects who had been shown the path from an egocentric perspective tended to use right angles to avoid the obstacle, whereas subjects who had been shown the path from an allocentric perspective preferred oblique-angled paths.

As previously described in Section 1.2, there are many environments where path planning is very difficult. In those situations, people use a variety of heuristics to arrive at their goal without a plan. First, people search for alternatives when faced with a navigation decision. In a virtual maze environment, a study found that participants followed three heuristic search strategies: enfilading, thigmotaxis, and visual scan (Kallai et al., 2005). *Enfilading* moves back and forth in a small area of the environment. *Thigmotaxis* stays near a large object or on the periphery of an open area during navigation as a safety mechanism in an unfamiliar environment. *Visual scan* remains in a fixed position and turns in place to examine the environment. Another study found that anxious human navigators used thigmotaxis (Kállai et al., 2009).

Once the alternatives are clear or a person has decided to stop searching, several heuristics are used to decide which alternative to select. In a study in a virtual indoor environment, subjects moved toward a landmark if they knew that the goal was close to that landmark (Waller and Lippa, 2007). A *beacon response strategy* triggers an action when the navigator sees landmarks that spatially correspond with movement along a learned path. For example, if on a previously travelled path a person turned at a distinctive statue, the beacon response strategy would indicate that next time you see the same statue you turn. A *survey-based strategy* uses learned survey knowledge about the spatial relations between objects in the environment to decide the direction towards a goal location when faced with a landmark. An early study in a virtual driving simulator showed that participants followed either a beacon response strategy, a survey-based strategy, or both (Aginsky et al., 1997). Another study in a virtual maze with landmarks showed that participants initially preferred a beacon response strategy but shifted to a survey-based strategy in later trials (de Condappa and Wiener, 2016). The *least-angle strategy* is a decision-making heuristic in which a person selects the direction most in line with the target's direction at an intersection in an unknown environment (Hochmair and Frank, 2000; Dalton, 2003). If the target is not visible, however, then a person estimates the position and direction of the target based on her perception and movement through the environment.

Spatial orientation has been used as a heuristic to determine the correct direction in which to travel in an unknown environment (Lawton, 1996). Two heuristics that maintain awareness of spatial orientation were observed in a study with two-year-old children: geometric reorientation and beacon guidance (Lee et al., 2012). Geometric reorientation reconciles the shape of the room as it is currently perceived with a representation of the room as it was previously experienced from a specific, known

direction. Beacon guidance uses the known spatial relationship between a landmark and the goal to infer spatial orientation.

People also use their frame of reference for decision making. One study examined the strategy used by subjects in a task to identify the direction of the initial position after navigation through a virtual tunnel (Gramann et al., 2005). Subjects showed preference for either egocentric or allocentric frames of reference to solve the task. They were also able to switch between reference frames and still accomplish the task. Another study also demonstrated that participants were able to use both frames of reference to navigate through a maze to a goal from different initial locations in the maze, and could switch between the strategies spontaneously (Iglói et al., 2009). That study also suggested that people can switch between the reference frames because they learn both representations simultaneously.

Several characteristics impact human navigation abilities and the heuristics they chose. A *sense of direction* is an awareness of one's location or orientation as an individual moves around the environment. It has been shown that individuals differ in their sense of direction and that these differences impact the heuristics they use in wayfinding (Cornell et al., 2003). Individuals with a good sense of direction were shown to use one of two strategies as they retraced a learned route through an outdoor environment: cardinal directions as a reference system, and landmarks to orient and localize (Kato and Takeuchi, 2003). The results also suggested that the same individuals could shift flexibly between strategies. Another study found that individuals with a good sense of direction learn routes in fewer trials and rely on visuospatial working memory (Baldwin and Reagan, 2009). A good sense of direction may allow individuals to minimize their localization error and thus improve their navigation abilities.

Navigational abilities and heuristics have also been observed to vary across demographics such as age and gender (Wolbers and Hegarty, 2010). One classic setting to evaluate navigational behavior is the *Morris Water Task* (MWT) in which an agent has to find a hidden platform in a circular arena (Morris et al., 1982). One study in a virtual MWT found that older adults prefer an egocentric strategy, whereas younger adults were evenly split between egocentric and allocentric strategies (Rodgers et al., 2011). Another study in a virtual environment found that older participants needed more time to form a cognitive map, and were slower and made more errors when they used their cognitive map (Iaria et al., 2009).

This section has reviewed the foundations of human reasoning and decision making. It has also discussed the kinds of spatial knowledge that people acquire during navigation and the internal representations they create with that knowledge. It described heuristics used by people for path planning and decision making during navigation without a plan. Finally, it summarized human factors that impact the heuristics used for navigation. The next section explores computational models for these human behaviors.

4.2 Cognitive Models of Human Navigation

Cognitive models seek to simulate observed human behavior with a computational system or algorithm. To produce the desired behavior, these simulations use such methods as logic, rules, determinism, and probability. Cognitive scientists then test them in artificial settings. For example, early work modeled how people plan as a series of decisions-making modules that opportunistically gave suggestions at different levels of abstraction (Hayes-Roth and Hayes-Roth, 1979). This work's paradigmatic approach presented a theoretical explanation for planning, proposed a cognitive model for the theoretical basis, executed the model on a computer, and then compared its

behavior with that of a human subject. This section discusses cognitive models of human navigation and reviews some autonomous robot navigation systems inspired by human behavior.

Early approaches simulated representations similar to cognitive maps. The TOUR model for multiple representations in a cognitive map incorporates route knowledge, path integration, and survey knowledge (Kuipers, 1978). TOUR found paths in a partially observable environment from an allocentric reference frame. Later work expanded TOUR into the Spatial Semantic Hierarchy (SSH) model (Kuipers, 2000). SSH models a cognitive map with hierarchical metric and topological representations. It also incorporates representations of partial knowledge and uncertainty. SSH has been implemented on simulated robots in indoor and outdoor environments and on a physical robot in an office environment. The Prototype, Location, and Associative Networks (PLAN) model also represented a cognitive map with a hierarchical structure, but from an egocentric perspective of the environment (Chown et al., 1995).

Some cognitive models have used a graphical approach to represent spatial knowledge. A formal logic-based model of the wayfinding process incorporated image schemata (recurring mental patterns that structure space) with affordances (what an object or environment enables people to do) into a wayfinding graph (a weighted, labeled directed graph of the state space) (Raubal and Worboys, 1999). A second model relied on route knowledge, where a route was a series of directed segments from one place to another and routes were connected to form a graph (Werner et al., 2000). A third study modeled navigation behavior with ACT-R, a general cognitive architecture that simulates human memory, information processing, and reasoning. It integrated a route-based representation to learn new environments, and a map-based representation to improve route following (Zhao et al., 2011).

A cognitive model for heuristic navigation decisions used a 2D virtual environment with static and dynamic obstacles (Gordon and Subramanian, 1997). It decomposed the overall task to reach the goal into the smaller tasks of either avoiding an obstacle or moving towards the goal, and it modeled each action's consequences for each smaller task. The model heuristically determines which smaller task the agent currently faces (i.e., whether or not the agent is close to an obstacle) and then selects an action to take for that task based on its action-consequence model. A follow-up study confirmed that people use a heuristic to recognize switches between tasks, and introduced a model of how that strategy shifts over time with experience in the environment (Gordon et al., 1998).

Biological and psychological explanations for human navigation can be useful starting points for autonomous robot navigation systems (Werner et al., 1997). One approach is to learn and use cognitive maps and other internal representations. Early work integrated a grid-based metric map with a topological map to adapt human-like internal representations of the environment for mobile robot navigation (Thrun, 1998b). The grid-based map, constructed with an artificial neural network, used Bayesian updating to determine the probability that a grid cell was occupied. To generate topological maps, the grid cells were grouped into connected regions. Thrun also adapted the human use of landmarks to guide navigation (Thrun, 1998a). In this work a mobile robot used a Bayesian approach to learn the position of landmarks in the environment, trained an artificial neural network to recognize the learned landmarks, and then used the landmarks for localization.

SemaFORR is a recent system that incrementally learns a spatial model during travel and uses commonsense qualitative spatial reasoning (Epstein et al., 2015). Its model relies on *spatial affordances*, abstract representations of the environment, to simulate a mental model similar to a cognitive map. SemaFORR has three kinds

of spatial affordances: regions (obstruction-free areas), trails (a shorter path derived from an actually travelled path), and conveyors (frequently visited areas of the environment). SemaFORR's decision-making mechanism combines multiple heuristics based on commonsense, the robot's sensor readings, and the spatial model to select an action.

This chapter has reviewed human heuristics for navigation and wayfinding. It discussed the internal representations (e.g., a cognitive map) that people create to reduce their cognitive load and the external cues in the environment that people use to make decisions. It also described cognitive models that computationally simulate human navigation and wayfinding behavior. Although some work has successfully modeled human navigation, a significant challenge for cognitive scientists is that it is difficult to account for individual differences. Despite this, behavioral experiments have gleaned some insights on human navigation behavior. This chapter has also reviewed several autonomous robot navigation systems inspired by human navigation behavior. Unlike metaheuristic methods for path planning that seek to find the shortest path in the least amount of time, these systems seek to exploit human knowledge and strategies to improve autonomous robot navigation. The next chapter suggests future work at the intersection of metaheuristics and cognitive models.

5. Conclusion

This paper has reviewed two significant research areas in autonomous robot navigation: metaheuristics and cognitive models. Most approaches are evaluated in simplified simulated environments that ease the hardware challenges of physical robots and reduce computational complexity. Ultimately, however, autonomous robot navigation systems must operate in the real world and contend with observability, multiagency, dynamism, continuity, and terrain. Future work must address these challenges in a real-world environment.

Metaheuristics are broadly applicable methods that use a heuristic strategy. Simulated annealing and tabu search are single-solution metaheuristics that use a heuristic to supervise the trade-off between exploration and exploitation. Although these algorithms have been adapted for path planning, they have not seen widespread use in the real world or sustained research in the literature.

Population-based metaheuristics have been used more frequently for path planning which suggests that they are able to find better solutions more efficiently than single-solution metaheuristics. Evolutionary algorithms use heuristics inspired by Darwinian principles, while swarm algorithms use heuristics inspired by the behavior of animals to search for optimal solutions. Both incorporate aspects of hill-climbing and randomization, and sometimes combine global search with local search. While some approaches have successfully navigated in both simulated and real-world environments, no single method has proved superior overall.

Population-based metaheuristics have several disadvantages. First, there is no guarantee that an optimal solution will be found in finite time, but this is not an issue in practice because in most cases a satisfactory solution suffices. Second, these methods can be computationally and memory intensive, although rapid advances in

hardware and its utilization partially offset this issue. Finally, the parameters of these approaches must be tuned by hand to enable good performance for path planning. This remains a significant issue and a major roadblock to widespread use of these approaches. To address it, there has been some work towards automated parameter tuning for heuristic algorithms, however, not specific to navigation or metaheuristics (Hoos, 2011).

Another approach to resolve the shortcomings of population-based metaheuristics has been to combine them with each other or with other methods. Although these hybrid metaheuristics draw upon the strengths of their components, it is difficult to compare them and identify the best because they have been tested with different numbers and types of robots in different and mostly simulated environments. No single hybrid metaheuristic has seen sustained research, which suggests that the field is still in its early stages and that many different methods will be tried and tested. At this point, significant research should be evaluated in real-world environments with physical robots to investigate the feasibility of actual deployment.

This survey also reviewed results from cognitive science that seek to explain and simulate human navigation and wayfinding. Studies show that people use both internal and external information and a variety of heuristics for decision making and reasoning during navigation. People also switch between these strategies within a task and across different tasks. Many metaheuristic methods must make simplifying assumptions about the environment to be able to plan paths successfully, even though people successfully navigate in the real world. Potential future work could synergize the research on metaheuristics and cognitive models to create more powerful autonomous robot navigation systems.

In conclusion, this paper has reviewed metaheuristic approaches for path planning and cognitive models of human navigation. Although there has been progress

towards autonomous robot navigation systems with these approaches, work remains to overcome their challenges. Potential future work could combine these approaches to produce a more robust system for autonomous robot navigation.

Appendix: Hybrid Metaheuristics

Metaheuristics in the tables are abbreviated as Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Bacterial Evolutionary Algorithm (BEA), Bat Algorithm (BA), Biogeography-based Optimization (BBO), Cuckoo Search (CS), Differential Evolution (DE), Evolutionary Programming (EP), Genetic Algorithm (GA), Gravitational Search (GS), Memetic Algorithm (MA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA).

Approaches	**Metaheuristics**	**Number of Robots**	**Robot Type**	**Environment**	**Evaluation**
Duan et al. (2010)	ACO & DE	Single	UAV	Static	Simulated
Wang et al. (2012c)	DE & CS	Single	UAV	Static	Simulated
Fu et al. (2013)	DE & PSO	Single	UAV	Static	Simulated
Roberge et al. (2014)	GA & PSO	Single	UAV	Static	Simulated
Wang et al. (2016)	DE & BA	Single	UAV	Static	Simulated
Châari et al. (2012)	GA & ACO	Single	Robot	Static	Simulated
Mo and Xu (2015)	PSO & BBO	Single	Robot	Static	Simulated
Contreras-Cruz et al. (2015)	ABC & EP	Single	Robot	Static	Simulated & Real-World
Botzheim et al. (2012)	MA & BEA	Single	Robot	Static	Simulated & Real-World
Das et al. (2016a)	DE & PSO	Multiple	Robot	Static & Dynamic	Simulated & Real-World
Das et al. (2016b)	PSO & GS	Multiple	Robot	Static & Dynamic	Simulated & Real-World

Table A1: Dual Metaheuristic Approaches

Approaches	**Metaheuristic**	**Other Method**	**Number of Robots**	**Robot Type**	**Environment**	**Evaluation**
Alvarez et al. (2004)	GA	Dynamic Programming	Single	Underwater Vehicle	Dynamic	Simulated
Arantes et al. (2016)	GA	Linear Programming	Single	UAV	Static	Simulated
Wu et al. (2006)	GA	Rough Sets	Single	Robot	Static	Simulated
Rakshit et al. (2013)	MA & DE	Reinforcement Learning	Multiple	Robot	Static	Simulated & Real-World
Masehian and Sedighizadeh (2010a)	PSO	Probabilistic Roadmap	Single	Robot	Static	Simulated & Real-World
Masehian and Sedighizadeh (2010b)	PSO	Probabilistic Roadmap	Single	Robot	Static	Simulated
Zhang et al. (2013)	PSO	Chaotic Search	Single	UAV	Static	Simulated
Xu et al. (2010)	ABC	Chaotic Search	Single	UAV	Static & Dynamic	Simulated
Cheng et al. (2014)	PSO	Chaotic Search	Single	UAV	Static & Dynamic	Simulated
Garcia et al. (2009)	ACO	Fuzzy Logic	Single	Robot	Static & Dynamic	Simulated
Li et al. (2006)	GA	Fuzzy Logic	Single	Robot	Static & Dynamic	Simulated
Kuo et al. (2016)	PSO	APF & Fuzzy Logic	Single	Robot	Static	Simulated & Real-World
Raja et al. (2015)	GA	APF	Single	Rover	Static	Simulated & Real-World
Miao et al. (2011)	GA	APF	Single	Robot	Static	Simulated
Cliff et al. (1993)	GA	Neural Network	Single	Robot	Static	Simulated
Gao and Tian (2007)	SA	Neural Network	Single	Robot	Static	Simulated

Table A2: Metaheuristic and Other Method Approaches

References

Aarts, H. and Elliot, A. (2012). *Goal-directed behavior*. Taylor & Francis.

Aginsky, V., Harris, C., Rensink, R., and Beusmans, J. (1997). Two strategies for learning a route in a driving simulator. *Journal of Environmental Psychology*, 17(4):317–331.

Ahmed, F. and Deb, K. (2013). Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Computing*, 17(7):1283–1299.

Alvarez, A., Caiti, A., and Onken, R. (2004). Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429.

Arantes, M. d. S., Arantes, J. d. S., Toledo, C. F. M., and Williams, B. C. (2016). A hybrid multi-population genetic algorithm for UAV path planning. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 853–860. ACM.

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA.

Baldwin, C. L. and Reagan, I. (2009). Individual differences in route-learning strategy and associated working memory resources. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 51(3):368–377.

Behnck, L. P., Doering, D., Pereira, C. E., and Rettberg, A. (2015). A modified simulated annealing algorithm for SUAVs path planning. *2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics*, 48(10):63–68.

Besada-Portas, E., de la Torre, L., Jesus, M., and de Andrés-Toro, B. (2010). Evolutionary trajectory planner for multiple UAVs in realistic scenarios. *IEEE Transactions on Robotics*, 26(4):619–634.

Besada-Portas, E., De La Torre, L., Moreno, A., and Risco-Martín, J. L. (2013). On the performance comparison of multi-objective evolutionary UAV path planners. *Information Sciences*, 238:111–125.

Bhattacharjee, P., Rakshit, P., Goswami, I., Konar, A., and Nagar, A. K. (2011). Multi-robot path-planning using artificial bee colony optimization algorithm. In *2011 Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 219–224. IEEE.

Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an international journal*, 8(2):239–287.

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373.

Blum, C., Puchinger, J., Raidl, G. R., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc.

Bongard, J. C. (2013). Evolutionary robotics. *Communications of the ACM*, 56(8):74–83.

Botzheim, J., Toda, Y., and Kubota, N. (2012). Bacterial memetic algorithm for offline path planning of mobile robots. *Memetic Computing*, 4(1):73–86.

Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82–117.

Brintaki, A. N. and Nikolos, I. K. (2005). Coordinated UAV path planning using differential evolution. *Operational Research*, 5(3):487–502.

Buniyamin, N., Sariff, N., Wan Ngah, W., and Mohamad, Z. (2011). Robot global path planning overview and a variation of ant colony system algorithm. *International journal of mathematics and computers in simulation*, 5(1):9–16.

Burchardt, H. and Salomon, R. (2006). Implementation of path planning using genetic algorithms on mobile robots. In *IEEE Congress on Evolutionary Computation*, pages 1831–1836. Citeseer.

Cai, Z. and Peng, Z. (2002). Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. *Journal of Intelligent and Robotic Systems*, 33(1):61–71.

Canny, J. (1988). *The complexity of robot motion planning*. MIT press.

Caponetto, R., Fortuna, L., Fazzino, S., and Xibilia, M. G. (2003). Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE transactions on evolutionary computation*, 7(3):289–304.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51.

Châari, I., Koubâa, A., Bennaceur, H., Ammar, A., Trigui, S., Tounsi, M., Shakshuki, E., and Youssef, H. (2014). On the adequacy of tabu search for global robot path planning problem in grid environments. *Procedia Computer Science*, 32:604–613.

Châari, I., Koubâa, A., Bennaceur, H., Trigui, S., and Al-Shalfan, K. (2012). smart-path: A hybrid ACO-GA algorithm for robot path planning. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

Chakraborty, J., Konar, A., Jain, L. C., and Chakraborty, U. K. (2009). Cooperative multi-robot path planning using differential evolution. *Journal of Intelligent & Fuzzy Systems*, 20(1, 2):13–27.

Cheng, Z., Wang, E., Tang, Y., and Wang, Y. (2014). Real-time path planning strategy for UAV based on improved particle swarm optimization. *Journal of Computers*, 9(1):209–214.

Chown, E., Kaplan, S., and Kortenkamp, D. (1995). Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1):1–51.

Cliff, D., Husbands, P., and Harvey, I. (1993). Explorations in evolutionary robotics. *Adaptive behavior*, 2(1):73–110.

Conlin, J. A. (2009). Getting around: making fast and frugal navigation decisions. *Progress in brain research*, 174:109–117.

Contreras-Cruz, M. A., Ayala-Ramirez, V., and Hernandez-Belmonte, U. H. (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing*, 30:319–328.

Cornell, E. H., Sorenson, A., and Mio, T. (2003). Human sense of direction and wayfinding. *Annals of the Association of American Geographers*, 93(2):399–425.

Dalton, R. C. (2003). The secret is to follow your nose: Route path selection and angularity. *Environment and Behavior*, 35(1):107–131.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410.

Darken, R. P. and Sibert, J. L. (1996). Wayfinding strategies and behaviors in large virtual worlds. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 142–149. ACM.

Das, P., Behera, H., Das, S., Tripathy, H., Panigrahi, B., and Pradhan, S. (2016a). A hybrid improved PSO-DV algorithm for multi-robot path planning in a clutter environment. *Neurocomputing*, 207:735–753.

Das, P., Behera, H., and Panigrahi, B. (2016b). A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm and Evolutionary Computation*, 28:14–28.

de Condappa, O. and Wiener, J. M. (2016). Human place and response learning: navigation strategy selection, pupil size and gaze behavior. *Psychological research*, 80(1):82–93.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

Dou, L., Li, M., Li, Y., Zhao, Q.-Y., Li, J., and Wang, Z. (2014). A novel artificial bee colony optimization algorithm for global path planning of multi-robot systems. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1186–1191. IEEE.

Duan, H., Yu, Y., Zhang, X., and Shao, S. (2010). Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm. *Simulation Modelling Practice and Theory*, 18(8):1104–1115.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.

Epstein, S. L., Aroor, A., Evanusa, M., Sklar, E. I., and Parsons, S. (2015). Learning spatial models for navigation. In *International Workshop on Spatial Information Theory*, pages 403–425. Springer.

Fogel, L. J. (1999). *Intelligence through simulated evolution: forty years of evolutionary programming*. John Wiley & Sons, Inc.

Fong, S., Deb, S., and Chaudhary, A. (2015). A review of metaheuristics in robotics. *Computers & Electrical Engineering*, 43:278–291.

Foo, P., Duchon, A., Warren, W. H., and Tarr, M. J. (2007). Humans do not switch between path knowledge and landmarks when learning a new environment. *Psychological research*, 71(3):240–251.

Foo, P., Warren, W. H., Duchon, A., and Tarr, M. J. (2005). Do humans integrate routes into a cognitive map? map-versus landmark-based navigation of novel shortcuts. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31(2):195–215.

Friedenberg, J. and Silverman, G. (2011). *Cognitive science: An introduction to the study of mind*. Sage.

Fu, Y., Ding, M., and Zhou, C. (2012). Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(2):511–526.

Fu, Y., Ding, M., Zhou, C., and Hu, H. (2013). Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(6):1451–1465.

Gallistel, C. R. (1990). *The organization of learning.* The MIT Press.

Gao, M. and Tian, J. (2007). Path planning for mobile robot based on improved simulated annealing artificial neural network. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 3, pages 8–12. IEEE.

Garcia, M. P., Montiel, O., Castillo, O., Sepúlveda, R., and Melin, P. (2009). Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing*, 9(3):1102–1110.

Gärling, T., Gärling, E., et al. (1988). Distance minimization in downtown pedestrian shopping. *Environment and Planning A*, 20(4):547–554.

Geng, N., Gong, D., and Zhang, Y. (2013). Robot path planning in an environment with many terrains based on interval multi-objective PSO. In *2013 IEEE Congress on Evolutionary Computation*, pages 813–820. IEEE.

Gigerenzer, G., Todd, P. M., ABC Research Group, t., et al. (1999). *Simple heuristics that make us smart*. Oxford University Press.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing*, 1(3):190–206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on computing*, 2(1):4–32.

Glover, F. W. and Kochenberger, G. A. (2003). *Handbook of metaheuristics*, volume 57. Springer Science & Business Media.

Golledge, R. G. (1999). Human wayfinding and cognitive maps. *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 5–45.

Gordon, D. and Subramanian, D. (1997). A cognitive model of learning to navigate. In *Proc. 19th Conf. of the Cognitive Science Society*, volume 25, pages 271–276.

Gordon, D., Subramanian, D., Haught, M., and Kobayashi, R. (1998). Modeling individual differences in learning a navigation task. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society: August 1-4, 1998, University of Wisconsin-Madison*, volume 20, pages 418–423. Lawrence Erlbaum Associates.

Graham, R. L. and Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57.

Gramann, K., Müller, H. J., Eick, E.-M., and Schönebeck, B. (2005). Evidence of separable spatial representations in a virtual navigation task. *Journal of Experimental Psychology: Human Perception and Performance*, 31(6):1199–1223.

Guan-Zheng, T., Huan, H., and Sloman, A. (2007). Ant colony system algorithm for real-time globally optimal path planning of mobile robots. *Acta Automatica Sinica*, 33(3):279–285.

Hayat, S. and Kausar, Z. (2015). Mobile robot path planning for circular shaped obstacles using simulated annealing. In *2015 International Conference on Control, Automation and Robotics (ICCAR)*, pages 69–73. IEEE.

Hayes-Roth, B. and Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive science*, 3(4):275–310.

Hernández, C., Asín, R., and Baier, J. A. (2015). Reusing previously found a* paths for fast goal-directed navigation in dynamic terrain. In *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pages 1158–1164.

Hillier, B. and Iida, S. (2005). Network and psychological effects in urban movement. In *International Conference on Spatial Information Theory*, pages 475–490. Springer.

Hirtle, S. C., Timpf, S., and Tenbrink, T. (2011). The effect of activity on relevance and granularity for navigation. In *Spatial Information Theory*, pages 73–89. Springer.

Hochmair, H. and Frank, A. U. (2000). Influence of estimation errors on wayfinding-decisions in unknown street networks–analyzing the least-angle strategy. *Spatial Cognition and Computation*, 2(4):283–313.

Hölscher, C., Büchner, S. J., Meilinger, T., and Strube, G. (2009). Adaptivity of wayfinding strategies in a multi-building ensemble: The effects of spatial structure, task requirements, and metric information. *Journal of Environmental Psychology*, 29(2):208–219.

Hölscher, C., Meilinger, T., Vrachliotis, G., Brösamle, M., and Knauff, M. (2006). Up the down staircase: Wayfinding strategies in multi-level buildings. *Journal of Environmental Psychology*, 26(4):284–299.

Hölscher, C., Tenbrink, T., and Wiener, J. M. (2011). Would you follow your own route description? cognitive strategies in urban route planning. *Cognition*, 121(2):228–247.

Hoos, H. H. (2011). Automated algorithm configuration and parameter tuning. In *Autonomous search*, pages 37–71. Springer.

Hu, Y. and Yang, S. X. (2004). A knowledge based genetic algorithm for path planning of a mobile robot. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04.*, volume 5, pages 4350–4355. IEEE.

Hussein, A., Mostafa, H., Badrel-din, M., Sultan, O., and Khamis, A. (2012). Metaheuristic optimization approach to mobile robot path planning. In *2012 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE.

Iaria, G., Palermo, L., Committeri, G., and Barton, J. J. (2009). Age differences in the formation and use of cognitive maps. *Behavioural brain research*, 196(2):187–191.

Iaria, G., Petrides, M., Dagher, A., Pike, B., and Bohbot, V. D. (2003). Cognitive strategies dependent on the hippocampus and caudate nucleus in human navigation: variability and change with practice. *Journal of Neuroscience*, 23(13):5945–5952.

Iglói, K., Zaoui, M., Berthoz, A., and Rondi-Reig, L. (2009). Sequential egocentric strategy is acquired as early as allocentric strategy: Parallel acquisition of these two navigation strategies. *Hippocampus*, 19(12):1199–1211.

Janzen, G., Schade, M., Katz, S., and Herrmann, T. (2001). Strategies for detour finding in a virtual maze: The role of the visual perspective. *Journal of Environmental Psychology*, 21(2):149–163.

Kala, R. (2012). Multi-robot path planning using co-evolutionary genetic programming. *Expert Systems with Applications*, 39(3):3817–3831.

Kalff, C. and Strube, G. (2009). Background knowledge in human navigation: a study in a supermarket. *Cognitive processing*, 10(2):225–228.

Kállai, J., Karádi, K., and Feldmann, Á. (2009). Anxiety-dependent spatial navigation strategies in virtual and real spaces. *Cognitive processing*, 10(2):229–232.

Kallai, J., Makany, T., Karadi, K., and Jacobs, W. J. (2005). Spatial orientation strategies in morris-type virtual water task for humans. *Behavioural brain research*, 159(2):187–196.

Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471.

Kato, Y. and Takeuchi, Y. (2003). Individual differences in wayfinding strategies. *Journal of Environmental Psychology*, 23(2):171–188.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks, 1995. Proceedings.*, volume 4, pages 1942–1948. IEEE.

Kennedy, J., Kennedy, J. F., Eberhart, R. C., and Shi, Y. (2001). *Swarm intelligence*. Morgan Kaufmann.

Khaksar, W., Hong, T. S., Khaksar, M., and Motlagh, O. R. E. (2012). Sampling-based tabu search approach for online path planning. *Advanced Robotics*, 26(8-9):1013–1034.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Klatzky, R. L. (1998). Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In *Spatial cognition*, pages 1–17. Springer.

Koenig, S. and Likhachev, M. (2002). D* lite. In *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pages 476–483.

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning a. *Artificial Intelligence*, 155(1):93–146.

Kok, K. Y. and Rajendran, P. (2016). Differential-evolution control parameter optimization for unmanned aerial vehicle path planning. *PloS one*, 11(3):e0150558.

Korf, R. E. (2014). Search: A survey of recent results. In *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, page 197. Morgan Kaufmann.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press.

Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive science*, 2(2):129–153.

Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial intelligence*, 119(1-2):191–233.

Kuo, P.-H., Li, T.-H. S., Chen, G.-Y., Ho, Y.-F., and Lin, C.-J. (2016). A migrant-inspired path planning algorithm for obstacle run using particle swarm optimization, potential field navigation, and fuzzy logic controller. *The Knowledge Engineering Review*, pages 1–17.

Latini-Corazzini, L., Nesa, M. P., Ceccaldi, M., Guedj, E., Thinus-Blanc, C., Cauda, F., Dagata, F., and Péruch, P. (2010). Route and survey processing of topographical memory during navigation. *Psychological research*, 74(6):545–559.

Lawton, C. A. (1996). Strategies for indoor wayfinding: The role of orientation. *Journal of environmental psychology*, 16(2):137–145.

Lee, S. A., Sovrano, V. A., and Spelke, E. S. (2012). Navigation as a source of geometric knowledge: Young childrens use of length, angle, distance, and direction in a reorientation task. *Cognition*, 123(1):144–161.

Leighton, J. P. (2004). Defining and describing reason. *The nature of reasoning*, pages 3–11.

Li, B., Gong, L., and Zhao, C. (2013). Unmanned combat aerial vehicles path planning using a novel probability density model based on artificial bee colony algorithm. In *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 620–625. IEEE.

Li, P. and Duan, H. (2012). Path planning of unmanned aerial vehicle based on improved gravitational search algorithm. *Science China Technological Sciences*, 55(10):2712–2719.

Li, Q., Zhang, W., Yin, Y., Wang, Z., and Liu, G. (2006). An improved genetic algorithm of optimum path planning for mobile robots. In *Sixth International Conference on Intelligent Systems Design and Applications*, volume 2, pages 637–642. IEEE.

Liang, J.-H. and Lee, C.-H. (2015). Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. *Advances in Engineering Software*, 79:47–56.

Loomis, J. M., Klatzky, R. L., Golledge, R. G., and Philbeck, J. W. (1999). Human navigation by path integration. *Wayfinding behavior: Cognitive mapping and other spatial processes*, pages 125–151.

Luke, S. (2013). *Essentials of Metaheuristics*. Lulu, second edition.

Ma, Y., Zamirian, M., Yang, Y., Xu, Y., and Zhang, J. (2013). Path planning for mobile objects in four-dimension based on particle swarm optimization method with penalty function. *Mathematical Problems in Engineering*, 2013:1–9.

Manikas, T. W., Ashenayi, K., and Wainwright, R. L. (2007). Genetic algorithms for autonomous robot navigation. *IEEE Instrumentation & Measurement Magazine*, 10(6):26–31.

Mansury, E., Nikookar, A., and Salehi, M. E. (2013). Artificial bee colony optimization of ferguson splines for soccer robot path planning. In *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 85–89. IEEE.

Masehian, E. and Amin-Naseri, M. R. (2008). Sensor-based robot motion planning-a tabu search approach. *Robotics & Automation Magazine, IEEE*, 15(2):48–57.

Masehian, E. and Sedighizadeh, D. (2010a). Multi-objective PSO-and NPSO-based algorithms for robot path planning. *Advances in electrical and computer engineering*, 10(4):69–76.

Masehian, E. and Sedighizadeh, D. (2010b). A multi-objective PSO-based algorithm for robot path planning. In *2010 IEEE International Conference on Industrial Technology (ICIT)*, pages 465–470. IEEE.

Meyer, J.-A., Husbands, P., and Harvey, I. (1998). Evolutionary robotics: A survey of applications and problems. In *European Workshop on Evolutionary Robotics*, pages 1–21. Springer.

Miao, Y.-Q., Khamis, A. M., Karray, F., and Kamel, M. S. (2011). A novel approach to path planning for autonomous mobile robots. *Control and Intelligent Systems*, 39(4):235–244.

Miller, G. A. (2003). The cognitive revolution: a historical perspective. *Trends in cognitive sciences*, 7(3):141–144.

Mittal, S. and Deb, K. (2007). Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 3195–3202. IEEE.

Mo, H. and Xu, L. (2015). Research of biogeography particle swarm optimization for robot path planning. *Neurocomputing*, 148:91–99.

Mohanty, P. K. and Parhi, D. R. (2016). Optimal path planning for a mobile robot using cuckoo search algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(1-2):35–52.

Morris, R., Garrud, P., Rawlins, J., and O'Keefe, J. (1982). Place navigation impaired in rats with hippocampal lesions. *Nature*, 297(5868):681–683.

Moscato, P. and Cotta, C. (2003). A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*, pages 105–144. Springer.

Panov, S. and Koceski, S. (2013). Harmony search based algorithm for mobile robot global path planning. In *2013 2nd Mediterranean Conference on Embedded Computing (MECO)*, pages 168–171. IEEE.

Pazzaglia, F. and De Beni, R. (2001). Strategies of processing spatial information in survey and landmark-centred individuals. *European Journal of Cognitive Psychology*, 13(4):493–508.

Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1):33–57.

Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. (2008). *A field guide to genetic programming*. Lulu. com.

Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

Qu, H., Xing, K., and Alexander, T. (2013). An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. *Neurocomputing*, 120:509–517.

Raja, R., Dutta, A., and Venkatesh, K. (2015). New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover. *Robotics and Autonomous Systems*, 72:295–306.

Rakshit, P., Konar, A., Bhowmik, P., Goswami, I., Das, S., Jain, L. C., and Nagar, A. K. (2013). Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multirobot path planning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):814–831.

Raubal, M. and Worboys, M. (1999). A formal model of the process of wayfinding in built environments. In *International Conference on Spatial Information Theory*, pages 381–399. Springer.

Richter, K.-F. and Winter, S. (2014). Cognitive aspects: How people perceive, memorize, think and talk about landmarks. In *Landmarks*, pages 41–108. Springer.

Roberge, V., Tarbouchi, M., and Allaire, F. (2014). Parallel hybrid metaheuristic on shared memory system for real-time UAV path planning. *International Journal of Computational Intelligence and Applications*, 13(02):1450008:1–16.

Roberge, V., Tarbouchi, M., and Labonté, G. (2013). Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Transactions on Industrial Informatics*, 9(1):132–141.

Rodgers, M. K., Sindone III, J. A., and Moffat, S. D. (2011). Navigation strategy as a predictor of navigation performance. In *Annual Meeting of Cognitive Science Society*, pages 2770–2775.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Sanders, G. and Ray, T. (2007). Optimal offline path planning of a fixed wing unmanned aerial vehicle (UAV) using an evolutionary algorithm. In *2007 IEEE Congress on Evolutionary Computation*, pages 4410–4416. IEEE.

Saska, M., Macas, M., Preucil, L., and Lhotska, L. (2006). Robot path planning using particle swarm optimization of ferguson splines. In *2006 IEEE Conference on Emerging Technologies and Factory Automation*, pages 833–839. IEEE.

Spiers, H. J. and Maguire, E. A. (2008). The dynamic nature of cognition during wayfinding. *Journal of environmental psychology*, 28(3):232–249.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3310–3317. IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.

Tavares, R. S., Martins, T., and Tsuzuki, M. d. S. G. (2011). Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning. *Expert Systems with Applications*, 38(4):2951–2965.

Thrun, S. (1998a). Finding landmarks for mobile robot navigation. In *1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings.*, volume 2, pages 958–963. IEEE.

Thrun, S. (1998b). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Tomko, M. and Richter, K.-F. (2015). Defensive wayfinding: Incongruent information in route following. In *International Workshop on Spatial Information Theory*, pages 426–446. Springer.

Tsai, C.-C., Huang, H.-C., and Chan, C.-K. (2011). Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. *IEEE Transactions on Industrial Electronics*, 58(10):4813–4821.

Tu, J. and Yang, S. X. (2003). Genetic algorithm based path planning for a mobile robot. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03.*, volume 1, pages 1221–1226. IEEE.

Tuncer, A. and Yildirim, M. (2012). Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38(6):1564–1572.

Tversky, B. (1993). Cognitive maps, cognitive collages, and spatial mental models. In *European Conference on Spatial Information Theory*, pages 14–24. Springer.

Waller, D. and Lippa, Y. (2007). Landmarks as beacons and associative cues: their role in route learning. *Memory & Cognition*, 35(5):910–924.

Wang, G., Chu, H. E., and Mirjalili, S. (2016). Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerospace Science and Technology*, 49:231–238.

Wang, G., Guo, L., Duan, H., Liu, L., and Wang, H. (2012a). A bat algorithm with mutation for UCAV path planning. *The Scientific World Journal*, 2012:1–15.

Wang, G., Guo, L., Duan, H., Liu, L., and Wang, H. (2012b). A modified firefly algorithm for UCAV path planning. *International Journal of Hybrid Information Technology*, 5(3):123–144.

Wang, G., Guo, L., Duan, H., Liu, L., Wang, H., and Wang, J. (2012c). A hybrid meta-heuristic DE/CS algorithm for UCAV path planning. *Journal of Information and Computational Science*, 5(16):4811–4818.

Werner, S., Krieg-Brückner, B., and Herrmann, T. (2000). Modelling navigational knowledge by route graphs. In *Spatial cognition II*, pages 295–316. Springer.

Werner, S., Krieg-Brückner, B., Mallot, H. A., Schweizer, K., and Freksa, C. (1997). Spatial cognition: The role of landmark, route, and survey knowledge in human and robot navigation. In *Informatik97 Informatik als Innovationsmotor*, pages 41–50. Springer.

Wiener, J., Ehbauer, N., and Mallot, H. (2009). Planning paths to multiple targets: memory involvement and planning heuristics in spatial problem solving. *Psychological Research PRPF*, 73(5):644–658.

Wiener, J. M., Lafon, M., and Berthoz, A. (2008). Path planning under spatial uncertainty. *Memory & cognition*, 36(3):495–504.

Wiener, J. M. and Mallot, H. A. (2003). 'Fine-to-coarse' route planning and navigation in regionalized environments. *Spatial cognition and computation*, 3(4):331–358.

Wiener, J. M., Schnee, A., and Mallot, H. A. (2004). Use and interaction of navigation strategies in regionalized environments. *Journal of Environmental Psychology*, 24(4):475–493.

Wolbers, T. and Hegarty, M. (2010). What determines our navigational abilities? *Trends in cognitive sciences*, 14(3):138–146.

Wu, C.-D., Zhang, Y., Li, M.-X., and Yue, Y. (2006). A rough set GA-based hybrid method for robot path planning. *International Journal of Automation and Computing*, 3(1):29–34.

Xu, C., Duan, H., and Liu, F. (2010). Chaotic artificial bee colony approach to uninhabited combat air vehicle (UCAV) path planning. *Aerospace Science and Technology*, 14(8):535–541.

Yamamoto, H. (1998). Robot path planning by genetic programming. *Artificial Life and Robotics*, 2(1):28–32.

Yang, L., Li, K., Zhang, W., Wang, Y., Chen, Y., and Zheng, L. (2015). Three-dimensional path planning for underwater vehicles based on an improved ant colony optimization algorithm. *Journal of Engineering Science and Technology Review*, 8(5):24–33.

Yang, X., Cai, M., and Li, J. (2016). Path planning for unmanned aerial vehicles based on genetic programming. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 717–722. IEEE.

Yao, Y., Ni, Q., Lv, Q., and Huang, K. (2015). A novel heterogeneous feature ant colony optimization and its application on robot path planning. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 522–528. IEEE.

Zhang, X. and Duan, H. (2015). An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Applied Soft Computing*, 26:270–284.

Zhang, Y., Wu, L., and Wang, S. (2013). UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization. *Mathematical Problems in Engineering*, 2013:1–9.

Zhao, C., Hiam, J. W., Morgan, J. H., and Ritter, F. E. (2011). A multi-strategy spatial navigation model in a text-based environment. In *Proceedings of the 20th Conference on Behavior Representation in Modeling and Simulation*, pages 251–258.

Zheng, C., Li, L., Xu, F., Sun, F., and Ding, M. (2005). Evolutionary route planner for unmanned air vehicles. *IEEE Transactions on Robotics*, 21(4):609–620.

Zhu, S. and Levinson, D. (2015). Do people use the shortest path? an empirical test of wardrops first principle. *PloS one*, 10(8):e0134322:1–18.