# BoulderSearch: Providing Personalized Bouldering Problem Solution Guided by Heuristic Search

Anonymous Submission to Research Methods Course BGU

March 2024

**Abstract**

Bouldering, a dynamic form of indoor climbing, demands effective problem-solving strategies to overcome the challenges presented by the climbing environment. This paper introduces **BoulderSearch**, a heuristic search framework that models bouldering problems as search problems. Leveraging dynamic movement sequences and heuristic search algorithms, BoulderSearch efficiently navigates through climbing holds to provide guidance for solving the bouldering problems. We propose a novel modeling framework for modeling bouldering problems as search problems while considering the individual climber's physical properties. In addition, we propose a novel heuristic based on our proposed modeling framework, which enables our proposed method to solve bouldering problems and provide solution guidance by explaining the search problem solution path. Our experimental results demonstrate that BoulderSearch can be applied to a wide range of bouldering problems and provides detailed visual solution guidance, making it a valuable tool for climbers and bouldering problem builders. Our code is available at https://github.com/IdanYankelev/ClimbingSolver

## 1 Introduction

Bouldering is a form of rock climbing that requires significant strength, endurance, and technique. It is a type of free climbing that typically involves small rock formations or artificial rock walls without the use of ropes or harnesses. The primary goal of bouldering is to climb to the top of a climbing wall by using a sequence of moves, known as a bouldering problem. These problems are usually less than six meters tall and require the climber to determine the best sequence of movements to climb the wall, taking into account factors such as wall features, hold placements, and the climber's physical abilities. Every bouldering problem must be started from the marked start holds, which are typically labeled with the word "Start" or identified by a tape of a different color than the rest of the holds on the problem. Once the climber has started the problem, they must work their way up the wall to the highest holds, which are typically the goal holds. These holds are usually the most challenging to reach, and they require a lot of strength, balance, and technique to hold onto.

There have been several solutions available to climbers in order to help them improve their skills and solve difficult bouldering problems by modeling these problems as graphs and applying search algorithms to efficiently compute their solutions by Katsura et al., 2021; Naderi et al., 2017; Pfeil et al., 2011; Stapel, 2023, however, some solutions provide non-interpretable results, making it difficult for climbers to understand how to improve their performance. Additionally, other solutions rely on fixed algorithms, which can be disadvantageous for climbers who possess unique physical attributes that could be used to their advantage.

Due to these limitations, many climbers feel that their individuality is not being taken into account when using existing digital solutions, as they believe that such solutions often provide a one-size-fits-all approach that is not tailored to their specific needs. Thus, there is a need for more personalized and interpretable solutions that are designed to take into account the unique physical attributes of each climber. By doing so, climbers can not only improve their skills but also gain a deeper understanding of how they can use their individual strengths to solve complex bouldering problems.

BoulderSearch closes this gap by modeling bouldering problems as search problems and solving them using personalized heuristics for each climber to better fit the individual climber's guidance. BoulderSearch shows effectiveness in solving various bouldering problems with different sets of physical

attribution, showcasing the generalization of our approach while providing clear visual guidance for solution steps to teach the climbers the needed steps to solve any feasible bouldering problem.

In summary, our contributions are as follows:

- We propose a novel generic modeling algorithm to model any bouldering problem with a climber's physical properties as a search problem.
- We propose a heuristic for the bouldering problem based on our proposed modeling algorithm, which allows for solving the bouldering problem as a search problem.
- We evaluate our approach to various bouldering problems, showing its generalization and adaptation to different problems and climbers' physical properties.

# 2 Background

## 2.1 Search Problem Modeling

Modeling a problem as a search problem involves defining states, actions, and goals to represent the problem as a graph to search for a path that solves the original problem. State, a node in the graph, represents a configuration of the problem at a given point in time. An initial state represents the problem's configuration at the beginning of the search, while goal states represent configurations where the desired problem solution is achieved. Actions are the transitions between states, therefore represented as edges in the graph and defined by the rules of the problem (how to transition between two configurations). By mapping the problem to this framework, search algorithms can systematically explore the state space to find a path from the initial state to a goal state, solving the problem. This approach is versatile and shown to be applicable for puzzles by Felner, 2015; Heij, 2023; Lewis, 2007; Stone and Stone, 1987, as well as for optimization problems by Askarzadeh, 2016; Civicioglu, 2013; Mahdavi et al., 2007, and for decision-making tasks in various domains by Fathollahi-Fard et al., 2023; Roijers et al., 2013.

## 2.2 Search Algorithms

Korf, 1985, presented various search algorithms in artificial intelligence, focusing on their application in tree searches, and discussed their complexities and effectiveness. All the search algorithms start at the root node of the search tree but search for the goal node using different rules, with the common one being BFS and DFS.

The Depth-First Iterative-Deepening (DFID) algorithm combines the advantages of BFS's completeness and space efficiency with DFS's depth penetration. It performs a series of depth-limited DFS, progressively increasing the depth limit with each iteration, as shown in Figure 1. This approach ensures that the space complexity is kept low while still guaranteeing the discovery of the shortest path. DFID is asymptotically optimal in terms of time, space, and solution path cost for exponential tree searches.

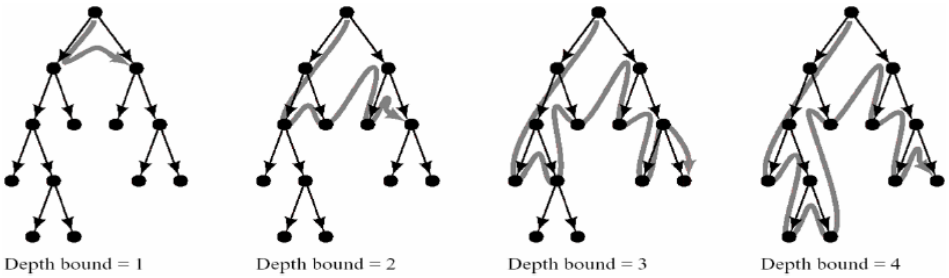


Figure 1: DFID example

Hart et al., 1968, introduced the A* search algorithm, which identifies the optimal path from a given source to a goal, assuming the heuristic is admissible. The A* algorithm is expressed as f(n) = h(n) + g(n), where g(n) represents the cost of the least expensive path from the start node to node n, as identified by A*, and h(n) denotes an estimated cost to reach a goal node from node n optimally. Another search algorithm covered by Korf [10] was the Iterative-Deepening A* (IDA*) algorithm which combines the iterative deepening strategy with A* to manage space complexity while ensuring the optimality of the solution. It iteratively increases the cost threshold for the depth-first search, effectively pruning the search tree and focusing on promising paths.

# 3 Related Work

Although the bouldering problem hasn't received much attention in the domains of search and path-finding problems, there have been some efforts to explore it in the literature.

Pfeil et al., 2011, addressed the challenge of designing climbing routes for the sport of free climbing. Traditionally, this process requires significant skill or extensive trial and error, making it difficult for novice climbers or route setters. The authors developed a software tool that allows users to design climbing routes by placing holds on a virtual wall. A key feature of this software was the simulated virtual climber that analyzes and visualizes the designed route. The software simulates the route using a path-planning algorithm and a virtual climber as the user adds, moves, or deletes holds. The main emphasis of their work was on the creation of the climbing wall, for example, the accessibility of how the wall is built (it describes how to work with the software to build a wall). Our research focuses more on solving an already existing bouldering problem and finding the best route given the climber's physical attributes, which were not considered in their work by assuming the same physical attributes for all climbers.

Naderi et al., 2017, addressed the challenge of path and movement planning for humanoid agents in bouldering, combining high-level path planning with low-level movement optimization to generate plausible climbing movements. Their technique employed a graph-based approach for path planning and sampling-based optimization for movement synthesis. Their Experiments demonstrated the system's ability to generate feasible climbing paths on various walls, with evaluation focusing on plausibility, CPU time, and the method's adaptability to climbers' physical capabilities. While this approach presented a much more complex body configuration representation and a higher range of possible body movements (e.g., feet on the wall and moving with both arms and legs at the same time), it was intended for humanoid agents and not for climbers, and therefore, lacked the interpretability our proposed method presents. In addition, while their method achieves solving the bouldering problem, it does not provide instructions on how to achieve the same solution, while our proposed method initially was aimed to be used as a guiding tool to teach new climbers as well as help bouldering problem builders to validate their problem's feasibility for a range of climber physical properties.

In a study conducted by Stapel, 2020, the main focus was on examining the potential of solving the bouldering problem using a greedy algorithm derived from an analysis of existing climbing routes. Stapel's algorithm used a heuristic based on classifying different holds and possible moves. For classifying climbing holds, Stapel considered their rotational angle relative to the ground, the difficulty of grasping a hold at its optimal rotational angle, and up to three types (primary type and two secondary types) as well as their difficulty levels ranging from 1 to 5 based on a survey conducted with experienced climbers to rate the difficulty of each hold. For classifying potential climbing moves, Stapel considered ten distinct moves and their difficulty based on an analysis of existing climbing routes done by experienced climbers. After creating these two classification systems, Stapel devised rules and formulas to determine the best next move and hold for each climbing position. Although his algorithm demonstrated effectiveness in solving the bouldering problem, it had a fixed nature and lacked flexibility for adapting to the individual needs of climbers. Our proposed method will utilize a dynamic algorithm considering each climber's physiological attributes. This algorithm leverages individualized physiological information to address and solve the bouldering problem effectively.

Katsura et al., 2021, delved into the complexity of pathfinding in climbing, focusing on how climbers

select holds and execute movements. It introduced a decision-making framework that evaluates potential holds and body positioning to navigate climbing routes effectively. While the research offers insights into climbing pathfinding strategies, it overlooked individual climber characteristics, such as physical capabilities. Our research aims to bridge this gap by customizing the heuristic search process to accommodate each climber's unique abilities. By tailoring our approach to individual climbers, we aspire to offer more personalized and effective solutions for climbers and route setters alike. This personalized approach could lead to more adaptable climbing strategies, addressing a significant limitation in the existing literature.

Stapel, 2023, extended his research in the following study, addressing the limitations of his previous research. The new approach used beam search, an optimization of best-first search (BFS) that reduces its memory requirements by limiting the number of candidates per level using a beam width factor. The beam search was initiated with all pairs of possible starting holds, and the search was finished once a hold on the top row was reached. The cost of path in the new approach was based on four parts, the distance between two holds, the difficulty of holding the hand-holds based on the individual's finger strength, how well the move matches previously seen climbing moves, and the difficulty of staying on the foot-holds. This new approach showed more flexibility in adapting to each climber's physiological attributes and demonstrated effectiveness in solving the bouldering problem; however, it only used the heuristic value and did not consider past information on the path (which is common behavior in BFS search and its optimizations), which resulted in several solutions that required using the individual's climbing capabilities to an extreme level. In our proposed method, we intend to implement the search using the A* algorithm to take advantage of past information (how difficult the path has been so far) to find optimal and considerable solutions to the climber's fatigue. In addition, while Stapel's solution only provided a final solution path to solve the bouldering problem, it was hard to interpret and not user-friendly for the climbing community. In our research, we provide a detailed visualization of the solution to allow a better understanding of the selected path.

# 4 Methodology

BoulderSearch finds an optimal solution for bouldering problems customized to the climber's physical properties. Given a bouldering problem and a climber's physical properties, BoulderSearch requires the following data: **1)** The coordinates of the bouldering problem holds. **2)** The coordinates of the starting holds. **3)** The coordinates of the finishing holds. **4)** The climber's arm length, from the shoulder joint to the hand joint, is referred to as ARM_LENGTH, measured in units of centimeters. **5)** The maximum angle between a climber's legs is referred to as LEG_SPREAD, measured in units of radians. **6)** The climber's leg length, from the hip joint to the foot joint, is referred to as LEG_LENGTH, measured in units of centimeters. **7)** The approximated coordinates of the shoulder, hip and knee joints when the climber holds the starting positions.

The pipeline of BoulderSearch is illustrated in Figure 2 and comprises the following steps: **1)** Our method requests the needed data. **2)** Our method models the given data into a search problem. **3)** Using the A* algorithm proposed by Hart et al., 1968, our method attempts to find a path from the starting search node to a goal node. **3.a)**If the A* algorithm returns a solution path, it is returned as the set of movements required to reach the finishing holds from the starting holds. **3.b)** Otherwise, our method prints that the given bouldering problem cannot be solved based on the climber's physical properties.
Sections 4.1-4.2 will provide a detailed description of each step in the processing pipeline.

## 4.1 Modeling the Bouldering Problem as a Search Problem

In the following section, we will detail the modeling process from a bouldering problem into a search problem, as illustrated in Figure 3. BoulderSearch's modeling process consists of three steps:
**1)** Modeling states representing the limb configurations in the current bouldering problem state.
**2)** Combining the limb states to represent the body configuration of the climber in the current boul-

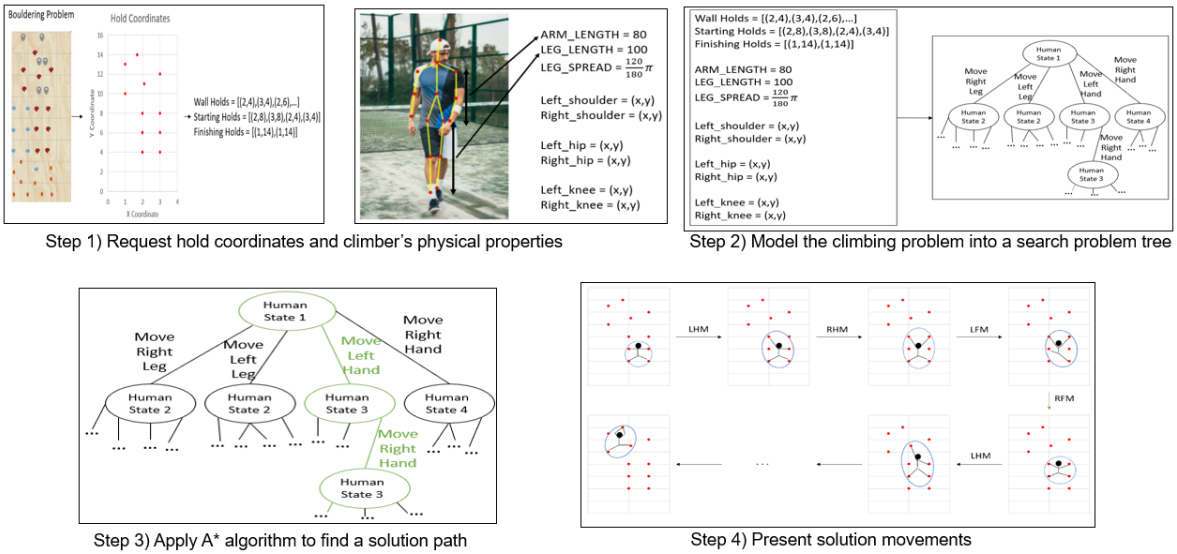


Figure 2: Proposed method steps

dering problem state.

**3)** Connecting human states if they are achievable from each other by moving a single limb.

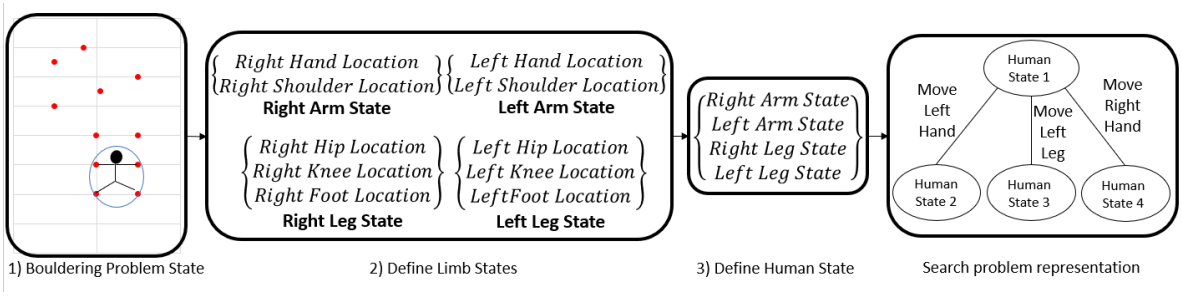


Figure 3: Modeling a bouldering problem as a search problem

First, given the starting hold locations and approximated joint locations, we initiate the problem modeling by modeling the starting state. Our modeling process begins with creating separate states for each limb. For the arm states, we consider the location of the hand and shoulder and verify that the Euclidean distance between these joints is less than the specified arm length. Moving on, we create the leg states by taking into account the location of the hip, knee, and foot and validating that the sum of distances between the hip and knee and the knee and foot is less than the specified leg length. Once we have confirmed the validity of each limb's status, we merge them to form a comprehensive representation of the climber's body configuration. We ensure that the angle between the legs is within the specified leg spread limit and that the line connecting the hips is parallel to the line joining the shoulders. This is an essential characteristic that will allow the mathematical modeling of limb movements in our search tree.

After obtaining the initial state, we proceed to discover its successors. Each successor is characterized by having only one hand/foot different from the previous state. We do this by trying to move each hand/foot to a new location on one of the holds in the problem, and then checking if the new state is valid. If the validity check returns positive, we know that the new state represents a new body configuration on the wall, in which only a single hand or foot has changed from the previous state, and that the movement was feasible. Due to space constraints, we are unable to provide a detailed explanation of the validity check's mathematical calculations. However, we will include a summary in the supplementary section. The process of finding successors is then repeated for each discovered state. We define the cost of moving between two states as 1. Therefore, the cost of moving between any two states can be defined as the sum of hand/foot movements required to reach from one state to

the other.

### 4.2 Solving the Search Problem Using the A* Algorithm

After obtaining a search problem model of the bouldering problem, we can now utilize search algorithms to solve it. Due to the problem's large state space (each state has the number of successors as the number of possible actions), we decided to use the A* algorithm and define the heuristic function as the expected hand movements from the current state to the goal state by dividing the distance between the hand locations in the current state and the goal state by the given arm_length:

$$\begin{aligned}\text{heuristic}(\text{human\_state}, \text{goal}) = &\frac{d(\text{human\_state.RightHandLocation}, \text{goal.RightHandLocation})}{\text{arm\_length}} \\ &+ \frac{d(\text{human\_state.LeftHandLocation}, \text{goal.LeftHandLocation})}{\text{arm\_length}}\end{aligned} \tag{1}$$

Based on the cost and heuristic definitions, we can ensure that the heuristic function is admissible. This is because it takes into account only the number of hand movements required to move from the current state to the goal state, but not the foot movements. Therefore, at any given time, we can see that h(state,goal)≤cost(state,goal).

Now that we have a model of the bouldering problem as a search problem, along with a cost function and a heuristic function, BoulderSearch uses the A* algorithm to attempt to solve it, as presented in Figure 3. If the algorithm returns a solution path, it will be a series of movements that can be made from the starting holds and end with the hands on the goal holds, using only legal climbing movements. The solution path can then be converted into a set of movement instructions for the climber to follow, allowing them to solve the bouldering problem by executing the same movements in the same order as depicted in the solution path, as can be seen in Figure 4.

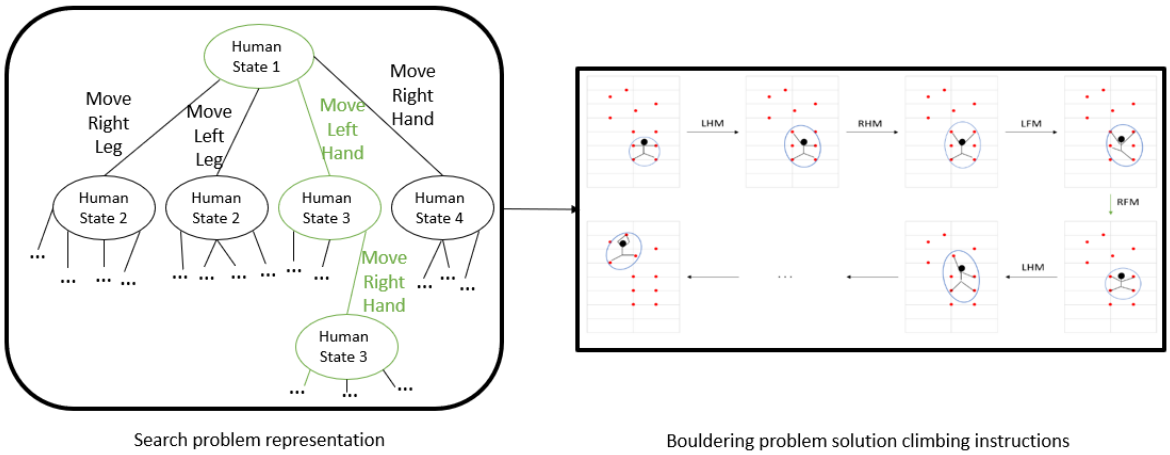


Figure 4: Converting search problem solution path to bouldering problem solution instructions

## 5 Experimental Setup

In our research, we aimed to determine whether a bouldering problem can be solved as a search problem; this question can be broken down into three sub-questions:

- Can a bouldering problem be modeled as a search problem?
- Can the search problem modeling consider an individual climber's physical properties?
- How similar will the movement order found by heuristic search and the movement order provided by an experienced climber be for the same bouldering problem?

For our approach implementation, we wrote code in Python using the Numpy package for mathematical calculations and the Heapq module for the A* implementation. We will use Numpy 1.22.0 and a GeForce RTX 3090 graphics card for the experiments.

In order to evaluate the effectiveness of our approach, we compare it to solutions provided by experienced climbers, as demonstrated in Stapel, 2020. We measure the similarity between the proposed solution and the solution provided by experienced climbers for the same problems by comparing the movement order and calculating the overlap percentage. To test the approach, we created a variety of bouldering problems with different wall sizes and hold distances, as well as multiple human models with varying arm and leg lengths, as presented in Figures 5-6.

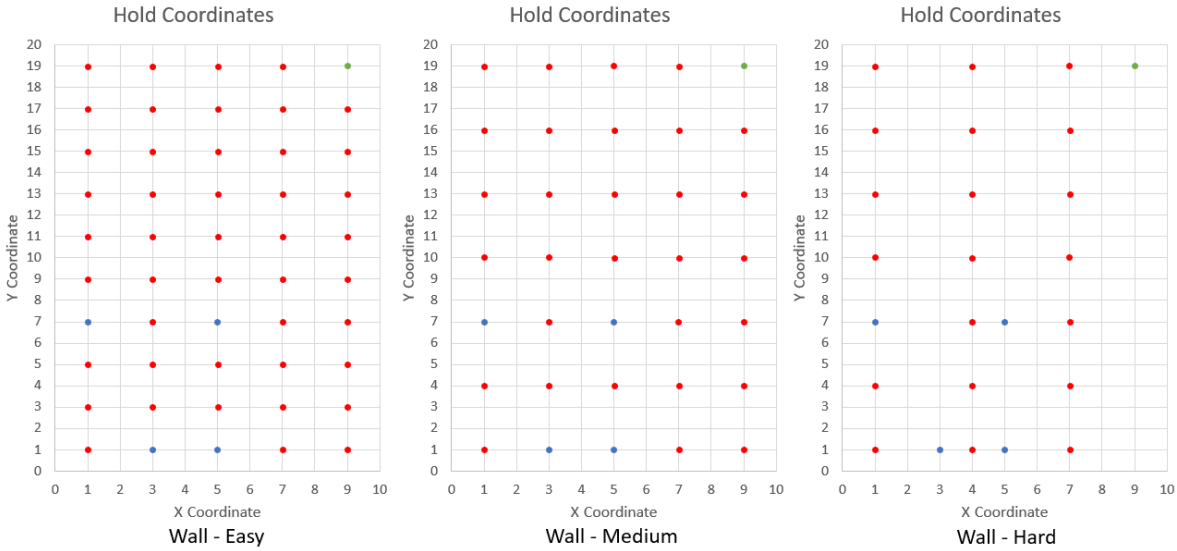


Figure 5: Wall modeling examples, starting holds colored in blue and goal hold colored in green

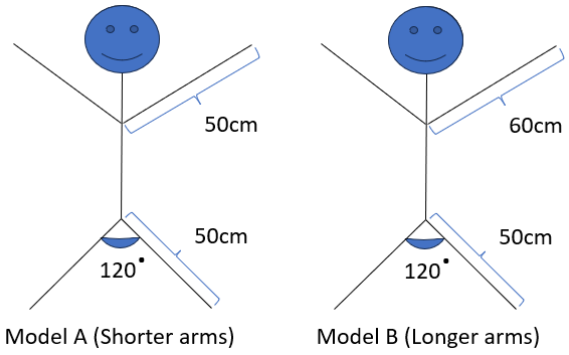


Figure 6: Human modeling examples

In our ablation study, we compare different heuristics with our modeling to evaluate the generalization of our modeling and the method's performance with different heuristic values to emphasize the importance of the heuristic. Furthermore, we plan to assess our proposed method on the same set of problems with different climbers' physical properties, such as ARM_LENGTH, LEG_LENGTH, and LEG_SPREAD. This comparison will help us understand how the solution is dependent on the set of climbers' physical properties used during the problem modeling phase.

# 6 Results

Table 1 compares the performance of BoulderSearch and an experienced climber on various bouldering problems of different wall sizes and difficulties. The table provides information about the size of each problem, the number of holds, and a comparison of the solutions provided by both BoulderSearch and the experienced climber. Table 2 presents the same experimental results but for a climber model with longer arms than the model in Table 1. These tables demonstrate that BoulderSearch effectively solved all of the problems. However, the overlap between BoulderSearch's solutions and those provided by the experienced climber was low. This could be attributed to various factors; one such factor could be the absence of gravity in our modeling process, which can result in optimal states for the climber that is impractical in reality due to the gravitational force exerted on the climber's body. Another factor could be a human error, which may lead to selecting a less efficient route than one BoulderSearch could identify. Another interesting observation compares the number of solution steps required for the two models. The climber in Table 2 needed fewer steps to solve the same problems by utilizing his advantage of longer arms, reaching further holds than the climber in Table 1.

Table 1: Comparison of climbing results with a climber with short arms (50 cm)

| Wall | Wall Size | Number of Holds | Number of Solution Steps by BoulderSearch | Number of Solution Steps by Experienced Climber | Overlap |
|---|---|---|---|---|---|
| easy | 10x20 | 50 | 17 | 15 | 35.3% |
| medium | 10x20 | 39 | 19 | 15 | 31.5% |
| hard | 10x20 | 25 | 22 | 16 | 45.4% |
| small | 8x14 | 28 | 10 | 9 | 50.0% |
| big | 20x25 | 66 | 27 | 24 | 26.0% |

Table 2: Comparison of climbing results with a climber with long arms (60 cm)

| Wall | Wall Size | Number of Holds | Number of Solution Steps by BoulderSearch | Number of Solution Steps by Experienced Climber | Overlap |
|---|---|---|---|---|---|
| easy | 10x20 | 50 | 16 | 15 | 37.5% |
| medium | 10x20 | 39 | 17 | 15 | 41.1% |
| hard | 10x20 | 25 | 19 | 15 | 42.1% |
| small | 8x14 | 28 | 9 | 9 | 55.5% |
| big | 20x25 | 66 | 25 | 22 | 36.0% |

In another experiment, with its goal to evaluate the effectiveness of BoulderSearch against unsolvable walls, our hypothesis was that BoulderSearch would fail to solve such walls, resulting in the absence of a solution path. As anticipated, BoulderSearch could not solve walls such as presented in Figure 7, which introduced large gaps that were impossible to bridge, leading to the classification of these walls as unsolvable after covering all the possible combinations of climbing moves.

In our ablation study, we aimed to evaluate BoulderSearch generalization to different heuristics as well as the importance of picking good heuristics for faster solutions. Table 3 presents a comparison between three heuristics: **1) Our Heuristic** - In this heuristic, we calculate the approximated number of hand movements between the current state and the goal states. **2) No Heuristic** - In this heuristic, we set the heuristic value to zero to demonstrate the performance when only past knowledge, such as in the DFS algorithm, is available. **3) Weighted Heuristic** - In this heuristic, we combined both our original heuristic as well as past knowledge of the effort to get to this state (the number of movements from the starting state and the current state), the weight was set to be 0.9 for the original heuristic and 0.1 for the past-knowledge. In this table, we compare the heuristics based on how many nodes each of them generated for the easy, medium, and hard walls from Figure 5 as well as the processing time to provide a solution.

The ablation results of two walls, the easy wall, and the medium wall presented in Figure 5, reveal an interesting finding. BoulderSearch solved the problem at a comparatively faster rate on both walls without using any heuristic in the easy wall and faster than using our heuristic in the medium wall. Additionally, the heuristic generated fewer nodes than the weighted heuristic or our heuristic in both walls. These results may have happened due to the simplicity of these walls, where a solution can

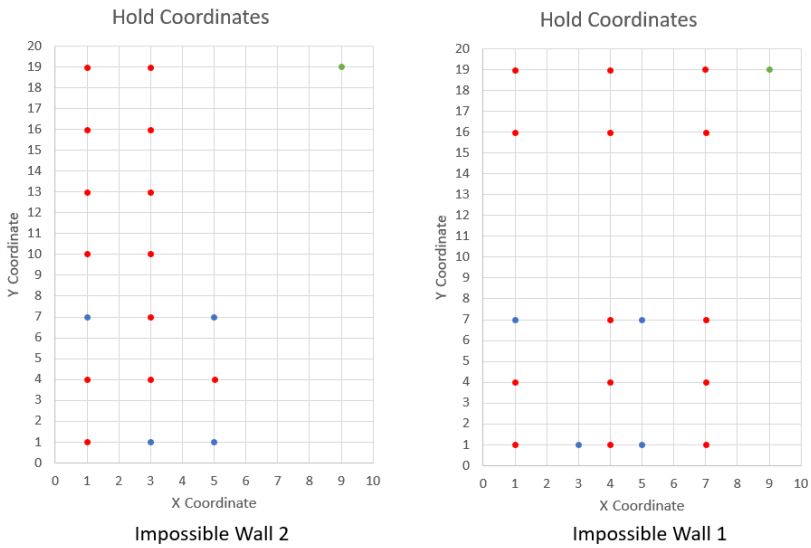


Figure 7: Unsolveable Walls, starting holds colored in blue and goal hold colored in green

Table 3: Heuristics comparison

| Heuristic | Easy Wall # of Nodes | Small Wall Solution Time [min] | Medium Wall # of Nodes | Medium Wall Solution Time [min] | Hard Wall # of Nodes | Hard Wall Solution Time [min] |
|---|---|---|---|---|---|---|
| No Heuristic | **37346** | **8.191353** | 96421 | 33.650821 | 7677 | **2.151025** |
| Weighted Heuristic | 37364 | 8.866349 | **95468** | **28.956932** | 7240 | 2.239033 |
| Our Heuristic | 37351 | 9.006123 | 95899 | 29.936404 | 7240 | 2.696656 |

be found by simply climbing in a straight line and then adjusting to the right, which might in some cases, require fewer moves than climbing in diagonally as BoulderSearch attempts with our heuristic (by minimizing the euclidian distance between the climbing states and the goal state). In the hard wall, our heuristic outperformed the search without any heuristic in terms of the number of generated nodes but still required a longer time to solve, which might have been caused by the aggregated time added to the solution by calculating the heuristic for all the generated nodes.

# 7 Conclusions

As discussed in the results section, our proposed method has proven effective in modeling the bouldering problem as a search problem. This has been achieved by defining a state as the climber's body configuration at a given time and connecting states that differ by only a single hand or foot movement. By doing so, we have been able to model a sequence of movements that enable the climber to climb the wall from the start to the goal. Our modeling considered the climber's physical properties to verify which states represent a feasible climber's body configuration and determine whether a movement between two states is possible. In our experiments, we showed that while the proposed method successfully solved a variety of bouldering problems, the solution paths were different from the paths chosen by experienced climbers, which may be due to the simple mathematical modeling chosen in this research, highlighting the fact that future work needs to be done to improve the mathematical movements as well as take into consideration other parameters as the strains of gravity on the chosen body states and fatigue of the climber while climbing the wall. After conducting this research, we showed that a bouldering problem can be solved by treating it as a search problem, providing a clear guide for solving the problem. This tool can be helpful for both climbers and bouldering problem builders by providing a visual demonstration to assist climbers in solving bouldering problems and learning how to improve their climbing. Additionally, bouldering problem builders can use the tool to validate the feasibility of their problems for different groups of climbers by providing the average physical attributes of the group to validate that the problem is solvable or needs to be adjusted to become solvable for the target group of climbers.

# References


Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & structures*, *169*, 1–12.

Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and computation*, *219*(15), 8121–8144.

Fathollahi-Fard, A. M., Wong, K. Y., & Aljuaid, M. (2023). An efficient adaptive large neighborhood search algorithm based on heuristics and reformulations for the generalized quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, *126*, 106802.

Felner, A. (2015). Early work on optimization-based heuristics for the sliding tile puzzle. *Workshops at the Twenty-Ninth AAAI Conf. on Artificial Intelligence*, 32–38.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

Heij, A. (2023). *Solving the 5x5x5 rubik's cube: Heuristic approach* [B.S. thesis]. University of Twente.

Katsura, D., Nishino, N., Sakamoto, D., & Ono, T. (2021). Climbing pathfinding with the holds and a decision method of the difficulty level of the holds. *International Workshop on Advanced Imaging Technology (IWAIT) 2021*, *11766*, 288–293.

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, *27*(1), 97–109.

Lewis, R. (2007). Metaheuristics can solve sudoku puzzles. *Journal of heuristics*, *13*(4), 387–401.

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied mathematics and computation*, *188*(2), 1567–1579.

Naderi, K., Rajam"aki, J., & H"am"al"ainen, P. (2017). Discovering and synthesizing humanoid climbing movements. *ACM Transactions on Graphics (TOG)*, *36*(4), 1–11.

Pfeil, J., Mitani, J., & Igarashi, T. (2011). Interactive climbing route design using a simulated virtual climber. In *Siggraph asia 2011 sketches* (pp. 1–2).

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, *48*, 67–113.

Stapel, F. (2020). *A heuristic approach to indoor rock climbing route generation* [B.S. thesis]. University of Twente.

Stapel, F. (2023). *Automated grade classification and route generation with affordances on climbing training boards* [Master's thesis, University of Twente].

Stone, H. S., & Stone, J. M. (1987). Efficient search techniques—an empirical study of the n-queens problem. *IBM Journal of Research and Development*, *31*(4), 464–474.

# A Supplementary

Algorithm 1 presents the pseudo-code for checking if a human state is feasible. The primary criterion for a feasible state is that the limb states must be possible. For arms, the length between the shoulder and the hand should be less than ARM_LENGTH, and for legs, the combined lengths from foot to knee and knee to hip should be less than LEG_LENGTH. Additionally, the angle between the hip-knee vector and the knee-foot vector should be less than 180.

After validating the individual limbs, we checked their integration by verifying that the legs were not crossed and the arms were above the legs. Also, the angle between the two legs should be less than LEG_SPREAD. If the state passes all these tests, it is considered a possible state representing a feasible body configuration.

**Algorithm 1** Check if Human State is Possible

```
function IsPossibleState
    if not left_arm.is_possible_state() OR not right_arm.is_possible_state() then
        return False
    end if
    if not left_leg.is_possible_state() OR not right_leg.is_possible_state() then
        return False
    end if
    if left_leg > right_leg then
        return False
    end if
    if left_arm.hand_position[1] < left_leg.foot_position[1] OR
            right_arm.hand_position[1] < right_leg.foot_position[1] then
        return False
    end if
    if CalculateAngleBetweenLegs(left_leg, right_leg) > LEG_SPREAD then
        return False
    end if
    return True
end function
```

Another important calculation is presented in Algorithm 2; when searching for successor states, we considered the need to straighten up with the legs/hands to reach further hold. To straighten up the legs, we calculated the approximate location of the center of body mass by utilizing the vectors of both legs. Using this location, we were able to calculate where the body joints should be moved to represent the body in a state of straightened legs (a state in which there are straight lines between each leg and the center of mass). For the arms, we used the ARM_LENGTH value to create a circle around the shoulder joint and then calculated the approximate location of the hand in a straightened state.

**Algorithm 2** Straighten Up

```
function StraightenUp
    old_left_hip ← left_leg_state.hip_position
    old_right_hip ← right_leg_state.hip_position
    center_of_mass ← calculate_center_of_mass()
    if center_of_mass == None then
        return
    end if
    UPDATE_HIPS_POSITION_UPON_STRAIGHTEN_UP(center_of_mass)
    UPDATE_SHOULDERS_POSITION_UPON_STRAIGHTEN_UP(old_left_hip, old_right_hip)
    UPDATE_HANDS_POSITION_UPON_STRAIGHTEN_UP()
    UPDATE_KNEES_POSITION_UPON_STRAIGHTEN_UP(center_of_mass)
end function
```