JC
FZL
AY
April 29, 2016

Cal Band Final Project Writeup

Introduction

The file calband_transition.m was made to help the Cal Band assign positions and marching routes to its members during a transition. The code was designed to save time and is significantly faster than working out a solution to a transition by hand. The function calband_transition is composed of three main functions called possibleFinalPositions, myfinalPosition, and StepList. Each of these functions is described in more detail below.

Methodology and Function Performance

When designing the function, our group decided that the best approach would be to first find the possible final positions for marchers, then assign those positions. By eliminating non-valid solutions, we hoped to reduce wait time.

We considered creating a function which would compute the radius of possible movements for each marcher, then plot each radius on the coordinate system of the transition. Spots that overlapped with desired final positions would be occupied by a marcher whose radius of possible movements overlapped with the target spot. We chose not to do this method because it was not very efficient: it calculated all the possible final locations for each marcher, even if those final locations were not a part of the target formation. As a result, we changed our approach to eliminate non-target locations from the field of final possible positions.

The main strengths of calband_transition are that the function consistently assigns valid final positions to marchers in a transition and has a low run time even for large transitions. While the function has the lowest run time when solving transitions with fewer members, the function can generally solve transitions in a couple seconds. Final positions are assigned to all band members, which can be extremely useful when planning a transition. The main weaknesses of calband_transition are that it does not always make the “common sense” choice of final positions and does not limit collisions. The function assigns marchers to final positions from a list of possible final positions that is not sorted by distance from the marcher. The function chooses one of the possible wait times and directions without further consideration. The function would not prioritize moving to a close target over moving to a target the maximum distance away, which could be a problem in an actual band performance. It also does not limit the number of collisions during a transition, which could also be problematic during a performance.

Function Description

calband_transition is composed of three main functions: possibleFinalPositions finds the set of final positions in the target formation a given marcher can reach in a given number of beats, myfinalPosition assigns each marcher a final position, and StepList finds the possible routes the marcher could have taken to get to the position assigned in myfinalPosition. calband_transition calls these functions and compiles the information found from StepList.

The function possibleFinalPositions computes possible final positions for each marcher. The function outputs a cell array that lists the coordinates of the target positions that each marcher can reach within the given number of beats. The function first finds the coordinates of each marcher given in the initial formation array, and stores their coordinates within an array while maintaining the numbering of marchers. Next, the function finds the coordinates of each target position and stores the coordinates in

another array. The function then finds the distance between each marcher and each target position by adding the absolute values of the differences between the coordinates of the initial positions and target positions. The target positions with distances less than or equal to the maximum number of steps allowed are then stored in the output cell array. The possible target coordinates for each marcher were stored in a cell array because it is ideal for storing double arrays with different dimensions. The run time was less than 1 second for all test cases.

The function myfinalPosition assigns each marcher a final position in the target formation. The function outputs a matrix containing the coordinates of the final position of each marcher. The function modifies the cell array calculated using possibleFinalPositions using a function called posSolutions. posSolutions calculates the number of possible final positions remaining for each marcher. myfinalPosition takes the information generated in posSolutions to find the marcher with the fewest number of remaining possible final positions and assigns that marcher a position on the target formation. Once a final position is assigned to a marcher, the marcher is removed from the set of marchers that have not been assigned final positions, and that final position is removed from the set of possible solutions for all other marchers. The function repeats until it has attempted to assign all band members a final position. The total run time of myfinalPosition varies depending on the number of marchers in the transition. For the largest test case, one iteration of the function took a little longer than five seconds. In the case that marchers are not assigned a valid final position, the list of final solutions is shuffled using the function changeInitial. The function is then called recursively with the new version of the cell array until the run time exceeds one minute. If no solution has been reached after one minute, the function will output the matrix of final values found in the previous iteration.

The function StepList finds the possible routes each marcher could take to get to their final position. The movements required in each direction were found by taking the difference between the position assigned to each marcher in myfinalPosition and their initial position for both the i and j directions. The output of StepList is a cell array containing both possible movement combinations for each marcher. Two movement combinations were possible because one direction change was allowed. The function takes less than a second to run.

The function calband_transition takes the instructions generated using StepList and interprets them in terms of compass directions and wait time and records the final position of the marcher in terms of i and j. The output of this function is a struct array with instructions for each marcher. Direction of movement is found from the order of movements and their direction found in StepList. Possible wait times are calculated by finding the difference between the maximum number of steps possible and the number of steps required to reach the target location. The final position is recorded in i and j space.

The overall run time of the function depends on the number of marchers in the transition and the number of recursive calls myfinalPosition must make in order to solve the transition. For the test cases given, the average time to complete the transition was around three seconds. The maximum possible run time for the function would be a little more than a minute.

Conclusion

Our Cal Band Final Project generally achieves the goal to help the Cal Band more efficiently find transition solutions with the calband_transition.m file assigning instructions to each Cal Band member with transition position and marching route, though more practical details need further improvements.