GIPP: General Interval Path Planning for Dynamic Environments

(I can't think of any better names…change to any name you feel more appropriate)

Author: Yu Hou

Last edit: 27/02/2020

Email: yhou0015@student.monash.edu

(Plz email me if you find any mistakes/better ideas/questions)

I. INTRODUCTION

This work is based on the SIPP (Safe Interval Path Planning) method, in which each location on the map is split into a sequence of safe/collision intervals along the timeline.

A safe interval is a continuous period of time for a location, during which there is no collision and it is in collision one timestep prior and after the period; whereas a collision interval is the opposite of a safe interval. Therefore, according to this definition, a safe interval should be bounded by collision intervals or infinity, so it is impossible for a robot to wait in place from one safe interval to another, because by doing so it will inevitably cross at least one collision interval.

The GIPP drops this limitation, by allowing a robot to cross collision intervals, with additional costs. It also introduces different costs (both temporal and spatial) for a robot to move between locations, which is set to 1 per grid for all time in the SIPP method.

Therefore, the GIPP method can be considered as a 'superset' to SIPP. According to the description, there is no need to know the location of obstacles for GIPP, the input will be a set of 'arrows', which represents the direction and value of costs.

II. ALGORITHM

Arrow:

An Arrow is the representation of costs. For instance, an obstacle locates at A at time *0* is to be moved to B at time *dt*, as shown in Figure 1. Then, arrows can be constructed as shown in the timeline.

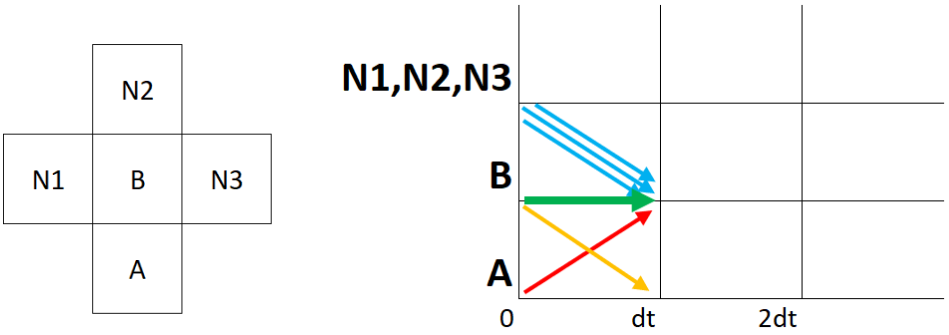

Figure 1 An example map and timeline, with an obstacle moving from A to B, (N1, N2 and N3 are neighbors of B)

The red arrow indicates the cost to move along the obstacle from A at time *0* to B at *dt*; The green arrow is the cost for the robot to wait in place at B from *0* to *dt*; The yellow arrow is the cost to move from B at *0* to A at *dt*, because by doing so the robot will collide with the obstacle en route; and finally, the 3 blue arrows are costs to move from any of the 3 neighbors at time *0* to B at *dt*, since the obstacle will arrive at B at time *dt*.

Therefore, for each step of an obstacle's path, there will be at most 6 arrows, they can all have different values. The green arrow can be termed as ***wait cost***, whereas the rest are ***transition costs***.

Treat wait cost as vertex cost:

To save the number of arrows, if the costs of green and blue arrows are equal, we can treat wait cost as a vertex cost, as show in Figure 2. Cost will be incurred for any path that passes location B at time *dt*. Consequently, the cost of red arrow should now be relative to the vertex cost, as opposed to the absolute cost before.

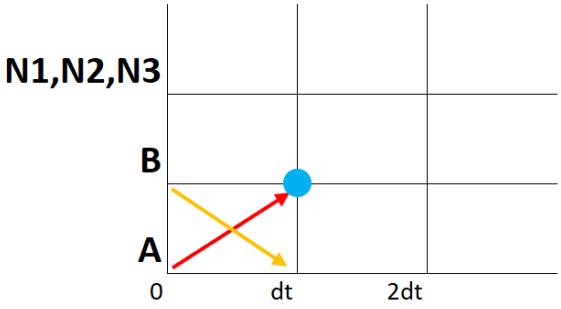

Figure 2 Treat wait cost as vertex cost

Store arrows:

For transition costs, an arrow is specified by "arrives at location *B* at time *t*, from location *A*, at a cost of *c*". Therefore, the red arrow is specified as (B,dt,A,cost), the yellow arrow is (A,dt,B,cost) etc.

For wait costs, an arrow is specified by "wait at location *B*, from time *t1* to *t2*, at a cost of *c*", so the green arrow is (B,[dt,dt],cost), and the example in Figure 3 is (B,[dt,2dt],cost).

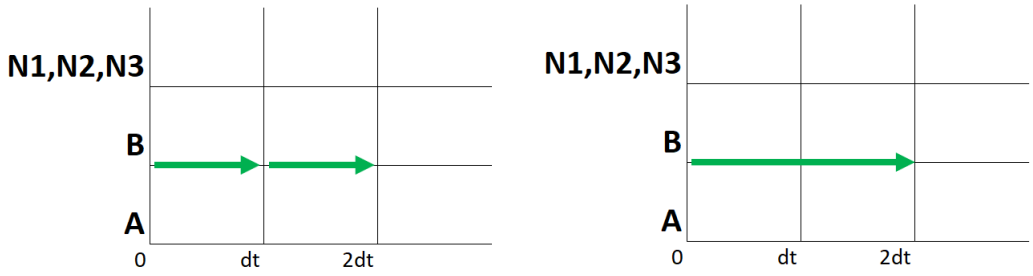

Figure 3 Wait cost arrows (left) can be combined into one arrow (right), if costs are same

It is important to note that only the end time of an arrow in recorded, since by doing so, wait cost will be same whether it is treated as vertex cost or edge cost; and it will be easier to convert arrows into timeline intervals.

Create Timeline (how to convert arrows into intervals as in SIPP):

Step 1: for a location, its interval is initialized as from 0 to infinity

Step 2: split the interval for each wait cost on this location

Step 3: split the intervals further based on transition costs pointed to this location.

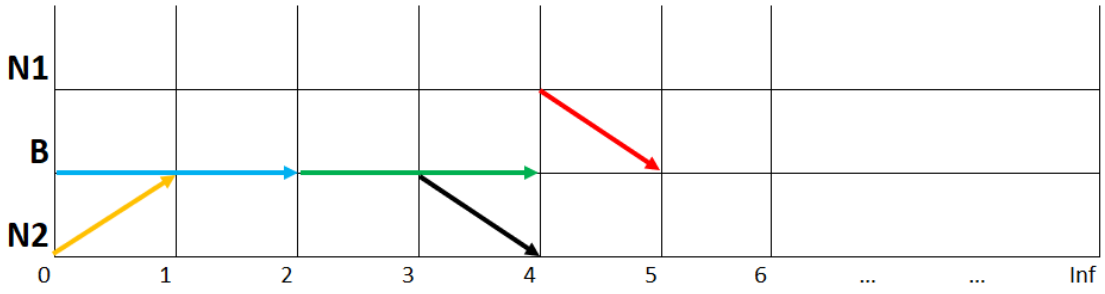

Figure 4 Example timeline. Different color means different costs.

As an example, for the timeline show in Figure 4:

Step 1: the interval of location B is initialized as [[0, Inf]]

Step 2: split the interval based on wait cost arrows: [[0, 0], [1, 2], [3, 4], [5, Inf]]

Step 3 split the intervals based on transition costs pointed to the location: [[0, 0], [1, 1], [2,2], [3, 4], [5, 5], [6, Inf]]

The operation can be completed in O(n) time, where n is the number of arrows, regardless whether arrows are sorted by time. The method is to read off all arrows at a location into a list in O(n) time, then build a heap directly from the list in O(n) time, and for each popped element, perform constant amount of operations to split intervals, since now we only need to split the last 1 or 2 intervals, instead of looking for the correct interval along the timeline to split.

Successors:

Similar to SIPP, we define a **state** by a location and an interval, and store time, cost, heuristics etc. as independent variables. The only major difference is that now intervals at the same location can be successors, as shown in Figure 5.

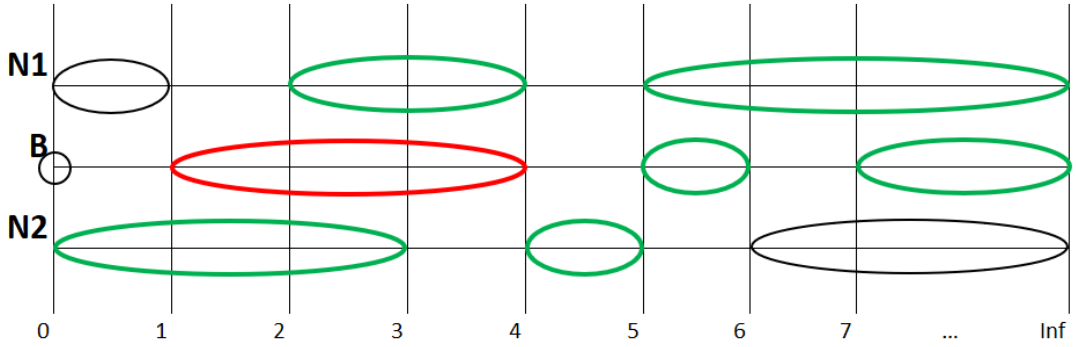

Figure 5 Green intervals are all successors to the red interval.

Cost between states:

Similar to SIPP, we use $c(s, s')$ to represent the cost to execute from state s to state s'. It is calculated as:

$$c(s, s') = cost\ to\ transition + transition\ cost + target\ vertex\ cost$$

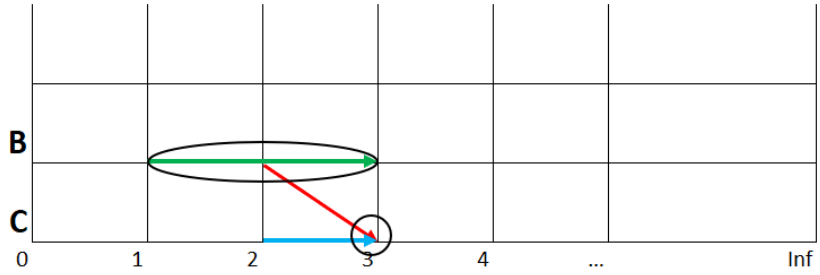

Figure 6 costs between states.

As shown in Figure 6, cost to transition is the wait cost arrow at B from time 1 to 2; transition cost is the value of the red arrow; and target vertex cost is the value of the blue arrow, if wait cost is to be treated as vertex cost, or 0 if wait cost is treated as edge cost.

A^* Termination criteria:

In SIPP, the termination criteria is when the priority queue is empty. We can, however, terminate earlier in GIPP. As shown in Figure 7, the robot starts at location (1,1) at time *0* and want to find an optimal path to location (2,4). There are 4 intervals at the goal location, but the first two all ends before time (2,4) - (1,1) = 4, which is the earliest possible time the goal can be reached, so there are 2 possible intervals at which the robot may arrive. Therefore, we can initialize a counter before A^*'s main loop, and for each state popped from the priority queue, we add 1 to the counter if its location is the goal, we can terminate when the counter equals to the number of possible intervals at the goal. This is true only when the heuristic is consistent, without this the approach loses completeness.

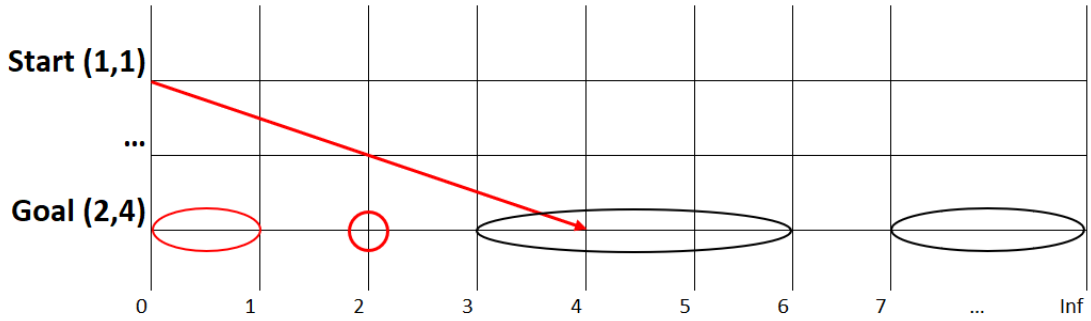

Figure 7 from (1,1) to (2,4). the red intervals can be omitted, since they end before the shortest amount of time the goal can be reached.

III. EXAMPLE

Example 1:

To show GIPP is a “superset” of SIPP, we first use an example that is covered in SIPP. As shown in Figure 8, two robots locate at (7,5) and (5,9) is moving leftwards and upwards respectively. The robot starts at (5,4) and the goal is (5,8).

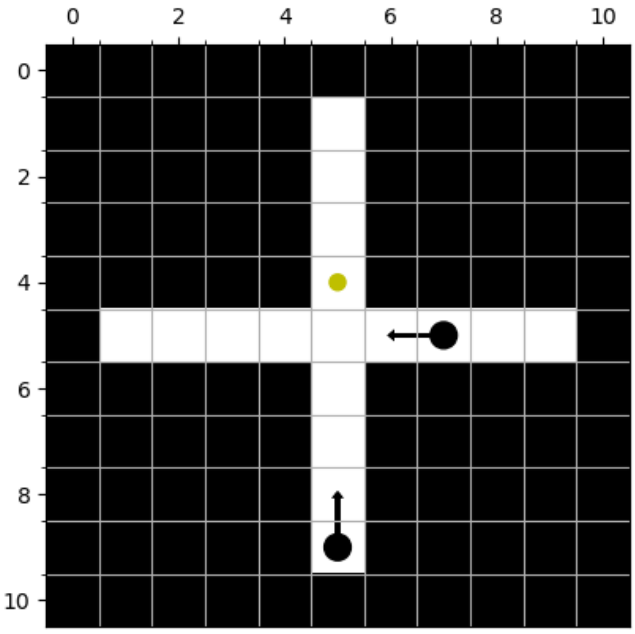

Figure 8 An example map with the robot locates at (5,4) and goal is (5,8).

Treat wait cost as vertex cost:

Since costs are all infinity, we can treat wait cost as vertex cost, to save the number of arrows. The solution is shown in Figure 9:

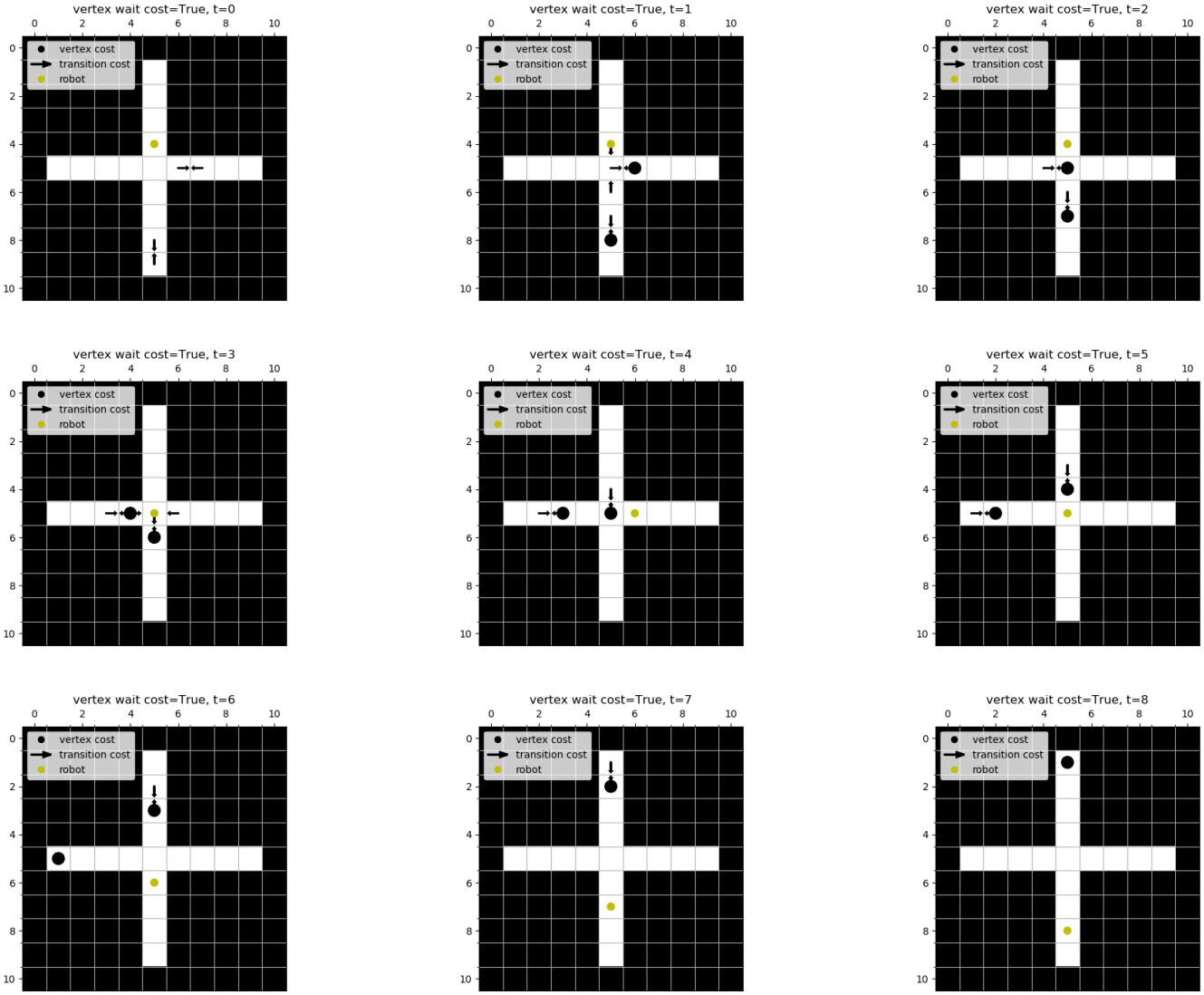

Figure 9 Example solution. Treat wait cost as vertex cost

Please be noted that the solution to this problem is not unique, there are multiple route with same costs.

The black circles are vertex wait costs, so robot can never reside on it, otherwise it will incur an infinite cost. A black arrow points from location A to B at time t means the cost will be infinite if the robot moves from A at time t to location B at time t+dt. For example, at time 1, there is a black arrow at the robot's location pointing downwards, this means the robot should not move downwards at time 1.

Treat wait cost as edge costs:

Alternatively, we can treat wait cost as edge costs, the solution is same, except the presentation of the solution is a bit different:

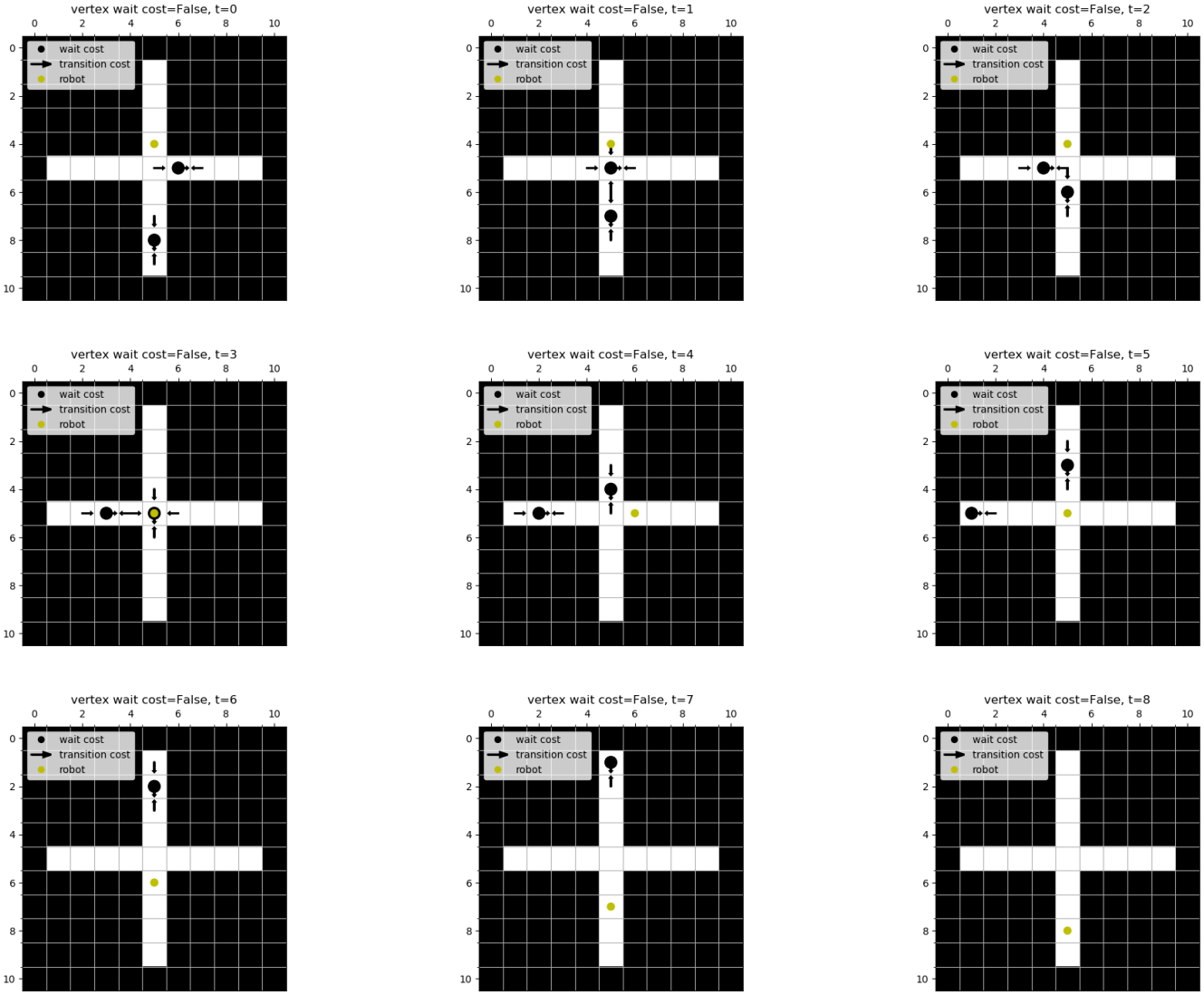

Figure 10 Example solution. Treat wait cost as edge cost

As shown in Figure 10, now there are more arrows. And now black circles are edge wait costs, they can be though of arrows pointing into the paper (or points along the time axis, if visualized in 3D). For example, at time 3, the robot resides on the black circle, it means if the robot chooses to wait in place, there will be an infinite cost, and based on the arrows around it, the robot can only choose to move rightwards or upwards.

Example 2:

A robot plans to move from location (1,1) to location (1,3), with different cost as shown in Figure 11. Here the wait cost is treated as edge cost for generalizability. By common sense, the blue line is the optimal path. The total cost is 207. The solution is also shown in Figure 12, in which arrow color indicates cost.

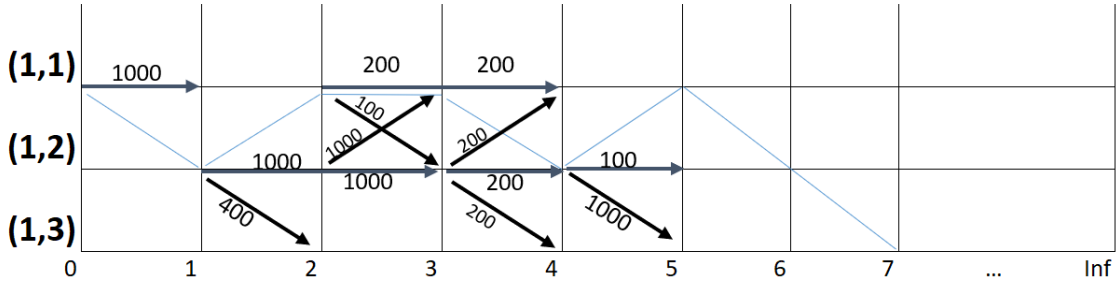

Figure 11 Example 2 timeline. Blue line is the optimal path.

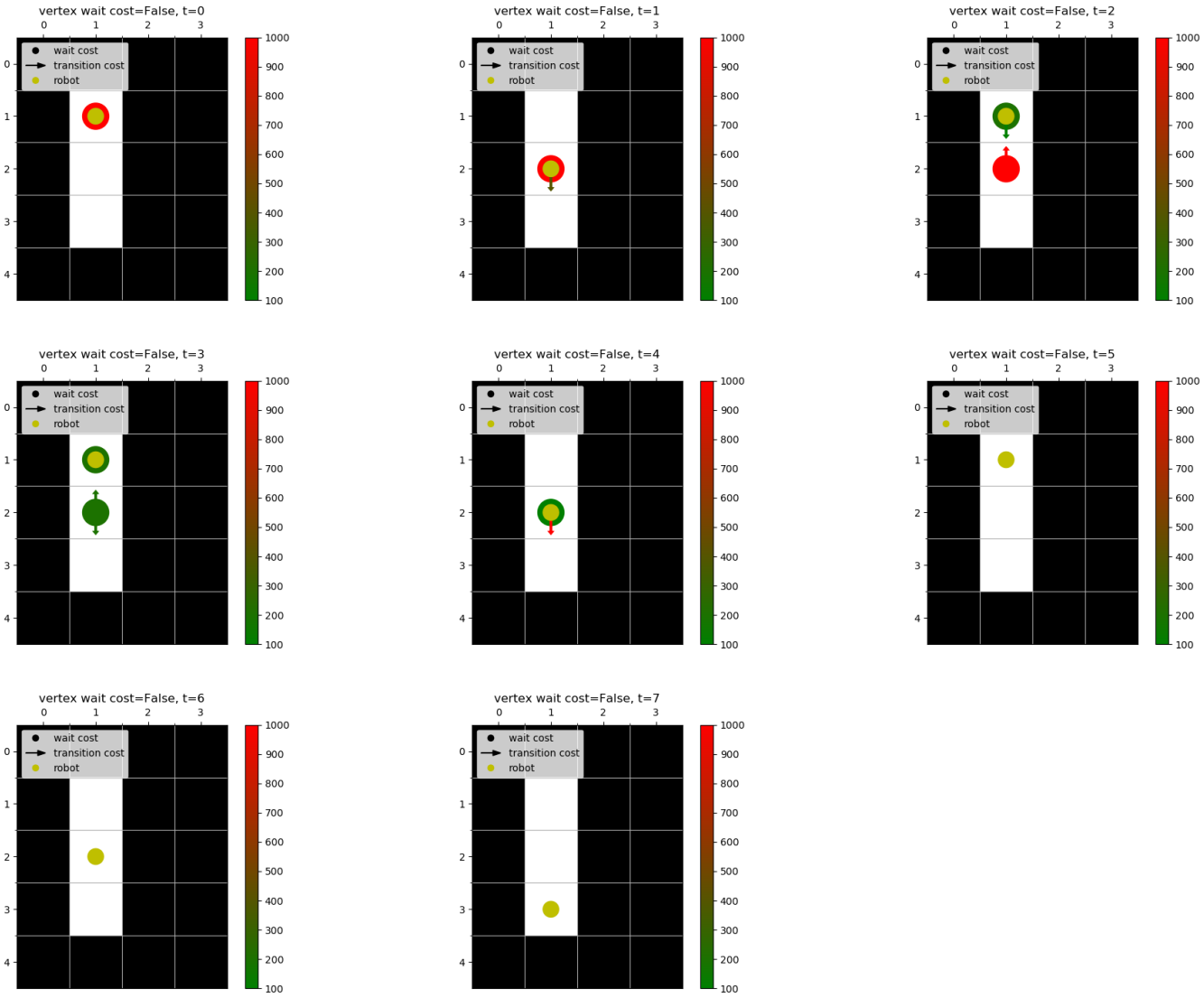

Figure 12 Solution to example 2. color indicates value