

Project 3: Another Version of Colored Trails

CMSC 421, Fall 2008

December 3, 2008

Due date/time: Monday, Dec 8, at noon

Late date/time: Wednesday, Dec 10, at noon

For this project, you will write an agent for a more complicated version of Colored Trails. The rules are like those in Project 2, with the following changes:

- If one player has finished but the other has not, the players can still make agreements and exchange chips.
- Whenever one agent makes a proposal, the other agent can accept, decline, or make a proposal of its own. Consequently, several proposals can pass back and forth before the agents reach an agreement.
- Instead of a path to follow, you will be given an undirected graph, a *start* node (your current node at the start of the game), and a *goal* node (the node you want to get to). Each node of the graph will have a color, and you must give up a chip of that color in order to go to that node.
- You don't have to move on each turn. For example, you might not want to move if your only possible move takes you further from the goal.

1 Setup, Play, and Scoring

As before, the names of the colors are positive integers. The node names also are numbers.

Setup. The game supervisor (the computer program that runs the game) initializes the game by choosing each of the following things for each agent:

- the agent's undirected graph.
- how many chips the agent has of each color,
- the agent's *synergy* parameter, which is a number between 0 and 1, inclusive,
- whether the agent is the proposer (see below) in the first turn.

Play. Each turn in the game consists of the following steps, which are repeated again and again until one or both agents finish. As in Project 2, the roles of *proposer* and *responder* are interchanged at each turn: if an agent is the proposer in the i 'th turn, then it will be the responder in the $i + 1$ 'th turn, and vice versa.

- **Bargaining.** The proposer may (but doesn't have to) make a proposal to trade chips with the responder. As in Project 2, an agent's proposal includes how many chips of each color the agent wants to give, and how many chips of each color the agent wants to get.

Whenever an agent makes a proposal, the other agent may accept it, reject it, or make a proposal of its own. The agents will continue making proposals to each other until one of them accepts or rejects the other's latest proposal.

- **Exchange.** This is the same as in Project 2. If a proposal was made and accepted, then each agent may either *honor* the agreement (i.e., give the other agent the agreed-upon numbers of chips) or *renege* (i.e., give the other agent nothing). Each agent must decide whether to honor or renege *before* finding out whether the other agent has honored or reneged.
- **Movement.** If an agent hasn't finished yet, then it may (but doesn't have to) move to any of the current node's neighbors,¹ by giving up a chip of the same color as the neighbor.

An agent can finish the game in two ways: if it reaches its goal, or if it can't move for three turns in a row. If both agents finish at the same time, then the game ends immediately. If one agent finishes before the other, then the bargaining, exchanges, and movements will continue until both agents have finished, except that the finished agent won't make any more movements.

Scoring. Scoring is exactly the same as in Project 2. Once both agents have finished, each agent's *partial score* depends on how many chips it has left, how many squares it traveled, and whether it reached its goal:

$$\text{partial score} = \begin{cases} \text{chips} + 3.0 * \text{squares}, & \text{if the agent reached its goal,} \\ \text{chips} + 1.5 * \text{squares}, & \text{if the agent didn't reach its goal.} \end{cases}$$

For example, if an agent has 10 chips left, and it moved 8 squares but didn't reach its goal, then its partial score is $10 + 1.5 * 8 = 22$.

An agent's *total score* is

$$\text{its partial score} + (\text{its synergy} * \text{the other agent's partial score}).$$

For example, if agent 0 has synergy 0.1 and partial score 20, and agent 1 has synergy 0.2 and partial score 10, the total scores will be:

$$\begin{aligned} \text{agent 0: } & 20 + 0.1 * 10 = 21 \\ \text{agent 1: } & 10 + 0.2 * 20 = 14 \end{aligned}$$

¹Nodes v and w are *neighbors* if the graph contains an edge $(v w)$ or $(w v)$.

2 Implementation

Implement your agent as a set of Allegro Common Lisp functions. You'll need the following functions for the game supervisor to call:

- (initialize *you proposer chips₀ chips₁ graph₀ graph₁ synergy₀ synergy₁*)

The supervisor calls this function to tell you to begin a new game with a new opponent.

- *you* is 0 if you are agent 0, or 1 if you are agent 1.
- *proposer* is 0 if agent 0 is the initial proposer, or 1 otherwise.
- *chips_i* (for $i = 0, 1$) is a vector $\#(q_1 q_2 \dots q_k)$, where k is the total number of colors, and each q_j is the number of chips of color j in agent i 's bag of chips.
- *graph_i* is agent i 's graph, in the following format:

$$(s\ g\ (c_1\ c_2\ \dots\ c_n)\ (v_1\ w_1)\ (v_2\ w_2)\ \dots)$$

where s is the start node, g is the goal node, each c_j is the color of node j , and each pair $(v_j\ w_j)$ is an edge between nodes v_j and w_j . Each c_j will be an integer in the range $1, \dots, k$, where k is the total number of colors. Each v_j or w_j will be an integer in the range $1, \dots, n$, where n is the total number of nodes. Each edge will appear at most once. Since the edges are undirected, v_j and w_j may be in either order.

- *synergy_i* is agent i 's synergy.

- (make-proposal)

The supervisor calls this function whenever you're the proposer. You should either return a proposal or NIL.

Like in Project 2, whenever you make a proposal, it should be a vector

$$\#(q_1\ q_2\ \dots\ q_k)$$

that tells how many chips of each color you want to give or get. For each j , if $q_j > 0$, this means you want to give q_j chips of color j to the other agent, and if $q_j < 0$, this means you want the other agent to give you $|q_j|$ chips of color j . If $q_j = 0$, it means you do not want to trade any chips of color j .

- (make-response *proposal*)

Unlike in Project 2, the supervisor only calls this function if the other agent has a proposal. *proposal* is the other agent's proposal, in the format described above. You can return T to accept the proposal or NIL to reject it, or you can return a proposal of your own.

- (give-chips *boolean*)

The supervisor calls this function when the bargaining step has ended. If *boolean* is T then there is an agreement between you and the other agent, and you should return T if you want to honor the agreement or NIL if you want to renege. If *boolean* is NIL then there is no agreement (i.e., either the proposer didn't propose anything, or the other agent rejected your proposal), and it doesn't matter what you return.

- (receive-chips *boolean*)
Like in Project 2, the supervisor only calls this function if there is an agreement between you and the other agent. *boolean* is T if the other agent is honoring the agreement, or NIL if the other agent is renegeing.
- (move-if-possible)
The supervisor calls this function at the start of the movement step. Unlike in Project 2, it will call this function even after you've finished (unless both agents are finished, in which case the game is over).

If you're not finished and *w* is a neighbor of your current node, then you can move to *w* if you have at least one chip of *w*'s color. If you move to *w*, you'll need to subtract 1 from your number of chips of *w*'s color.

Regardless of whether you move, you should return a list (*node chips*), where *node* is the node you're at, and *chips* is a vector $\#(q_1 q_2 \dots q_k)$ telling how many chips you have of each color.
- (their-position *node chips*)
The supervisor calls this function at the end of the movement step, to let you know the other agent's position. *node* and *chips* are the values that were returned by the other agent's move-if-possible function.

3 Grading

The grading criteria are the same as in Project 2:

- 20% on programming style (e.g., see the “programming style” link on the class page).
- 20% on documentation. In addition to commenting your code, write a short report (in either PDF or plain-text format) explaining how your program decides what proposal to make, how it decides whether to accept a proposal, and how it decides whether to renege. Include pseudocode.
- 50% on correctness. This will include things like whether your program keeps correct records of where it is and what chips it has left, whether it makes legal proposals and responses (e.g., you can't propose to give the other agent 5 chips of color 1 if you only have 4 chips of that color), whether it causes runtime errors, etc.
- 10% on performance (the score you get in a tournament that involves all of the programs in the class).