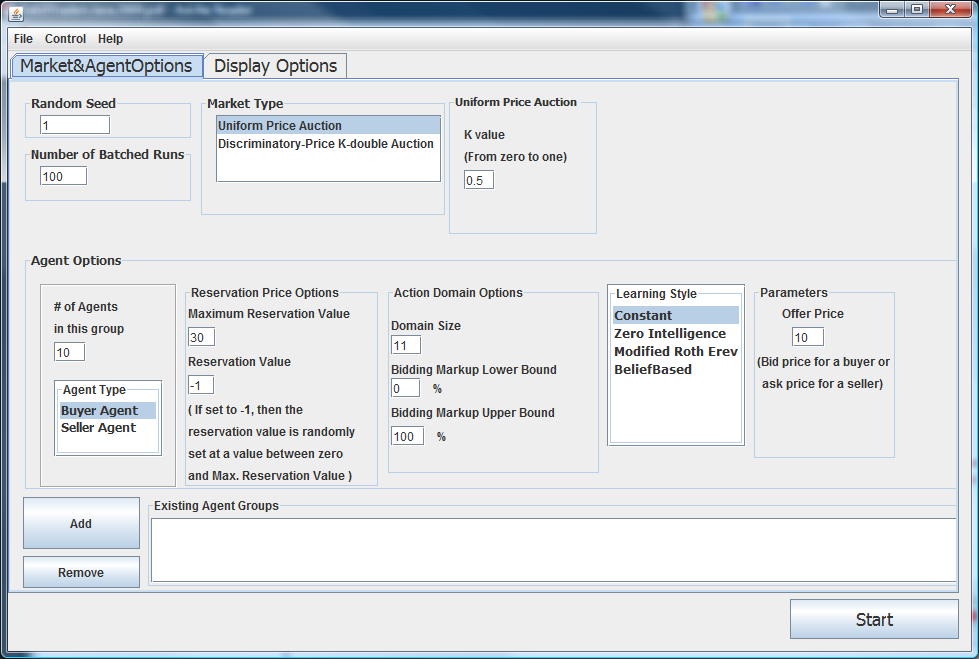
Part B)

**OPTION 2: Verify the DA-MAOS Code for MRE Learning Traders**

* VERIFY whether or not the DA-MAOS demo [1] permits the user to designate any number of seller traders (from 0 to NS) and buyer traders (from 0 to NB) to be learning traders.

Yes. The DA-MAOS demo allows you to add any number of Sellers or Traders, choosing which type of learning style for each of them. This can be seen on the lower portion of the GUI. First pick the agent type and how many you want to add to the group. You can then specify a learning style for that group, and add it. This allows for any combination of different kinds of learning styles and the number of buyers and sellers.

* Using your analysis in Part A, VERIFY whether or not the DA-MAOS demo [1] correctly codes for MRE reinforcement learning when “ModRothErev” is selected to be the learning method of a trader.

//TODO: Not quite sure what our analysis from part A was. I haven’t seen this doc yet.

* VERIFY whether or not the DA-MAOS demo [1] has a “stopping rule” that permits a user to stop a simulation run after a certain number M of time steps (or “ticks”).

The user (the person running the code), is able to stop or pause the test at anytime using the RepastJ toolbar. We added functionality to be able to end after a configurable # of rounds, seen below. The number of batches can also be changed in the GUI.

Added following code to TradingWorld.java to allow the number of configurable rounds.

//add action to end after configurable number of rounds

allActions.createActionFor(this, "checkEndCondition");

public void checkEndCondition()

{

//-1 is considered "infinity" (don't ever stop)

if(simSpec.getNumRounds() != -1 && clockTick >= simSpec.getNumRounds())

this.stop();

}

Fixed spelling error in *ModRothErev.java,*

Changed fi to phi ( the recency parameter in the Roth-Erev algorithm) in lines 19,48, 88 and 95.

Fixed a typo in *ModRothErev.java,* seen below.

//return reservationValue + ( (maxSellerValue - reservationValue)\* pickedAction );

// return (float) reservationValue + (reservationValue \* (myAdp.getLowerBound()

//+ pickAction() \* (myAdp.getUpperBound() - myAdp.getLowerBound() ))) /100;

return pickedAction \* reservationValue + reservationValue;

Removed:

else

throw new RuntimeException( " pickedAction = " + pickedAction +" which is not within the interval ( 0, 1 )");

We changed the GUI layout to include the number of rounds.

Added the following code to *AuctionMarketGUI.java* to do this:

Line 16: import javax.swing.GroupLayout;

Line 18: import javax.swing.LayoutStyle;

Line 144: numRounds = new javax.swing.JTextField();

Line 145: numRoundsLabel = new javax.swing.JLabel();

Line 603: numRoundsLabel.setText("Number of Rounds");

Line 604: numRounds.setText("150");

Lines 609 to 622:

jPanel9Layout.createParallelGroup()

.addGroup(GroupLayout.Alignment.LEADING, jPanel9Layout.createSequentialGroup()

.addComponent(numRoundsLabel, GroupLayout.PREFERRED\_SIZE, 106,

.addContainerGap(54, Short.MAX\_VALUE))

.addGroup(jPanel9Layout.createSequentialGroup()

.addPreferredGap(numRoundsLabel, numOfMarkets, LayoutStyle.ComponentPlacement.INDENT)

.addGroup(jPanel9Layout.createParallelGroup()

.addGroup(GroupLayout.Alignment.LEADING, jPanel9Layout.createSequentialGroup()

.addComponent(numOfMarkets, GroupLayout.PREFERRED\_SIZE, 48, GroupLayout.PREFERRED\_SIZE)

.addGap(0, 11, Short.MAX\_VALUE))

.addGroup(jPanel9Layout.createSequentialGroup()

.addComponent(numRounds, GroupLayout.PREFERRED\_SIZE, 59, GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap(89, 89)));

Lines 624 to 630:

jPanel9Layout.createSequentialGroup()

.addComponent(numOfMarkets, GroupLayout.PREFERRED\_SIZE, GroupLayout.PREFERRED\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 0, GroupLayout.PREFERRED\_SIZE)

.addComponent(numRoundsLabel, GroupLayout.PREFERRED\_SIZE, GroupLayout.PREFERRED\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 0, GroupLayout.PREFERRED\_SIZE)

.addComponent(numRounds, GroupLayout.PREFERRED\_SIZE, 21, GroupLayout.PREFERRED\_SIZE)

.addGap(0, 6, Short.MAX\_VALUE)

Line 1465: simSpecs.setNumRounds( Integer.valueOf(numRounds.getText()));

Line 1683: private javax.swing.JTextField numRounds;

Line 1684: private javax.swing.JLabel numRoundsLabel;

Added the following code to *SimSpecs.java* for use with the GUI interface.

Line 22: private int numRounds;

Line 30: numRounds = -1;

Lines 102 to 110:

public int getNumRounds()

{

return numRounds;

}

public void setNumRounds(int numRounds)

{

this.numRounds = numRounds;

}

Changed the following code in *RandomNumbers.java:*

This problem was addressed in exercise 5, which we found the range to be incorrectly specified.

From: // specifically [low, high]

To: + // specifically [low, high)

So, changed the following in *DiscPriceKDoubleAuction.java*

From:

- buyerIndex = rand.getRangedInt( 0, buyers.length -1 );

- sellerIndex = rand.getRangedInt( 0, sellers.length -1 );

To:

+ buyerIndex = rand.getRangedInt( 0, buyers.length );

+ sellerIndex = rand.getRangedInt( 0, sellers.length );