THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

COMP3702 Assignment 2

Semester 2 2015

Joshua Rillera (43150621)
Thomas Millward (43236945)

Problem Formulation

The state space S used in this solution is defined as follows:

$S = \{(x_0, x_1, ..., x_{m-1}) \mid x_0, x_1, ..., x_{m-1} <= y, x_0 + x_1 + ... + x_{m-1} <= z\}$

Where m is the number of different items, y is the maximum allowed number of each item and z is the capacity of the fridge . Put simply it is the set of all possible combinations of items in the fridge such that the constraints on the maximum number of any item are not violated.

The action space A is:

$A = \{(x_0, x_1, ..., x_{m-1}) \mid x_0, x_1, ..., x_{m-1} <= y, x_0 + x_1 + ... + x_{m-1} <= n\}$

Where m is the number of different items, y is the maximum allowed number of each item and n is the maximum number of items we can order. So A is the set of all possible purchases that can be made by the system such that we don't purchase more than is allowed and a given purchase won't cause the fridge to go over capacity.

At the beginning of each week, given the current inventory of our fridge, we purchase food and at the end of the week, we are left with the amount for each type of food after consumption. From this we defined the transition function as:

$T(s, a, s') = P(S_{t+1} = s' \mid S_t = s, A_t = a)$

where s, s' are elements of S and a is an element of A.

Given the current state, the reward will be the penalty in the current week multiplied by the discount factor, which is raised to the power of the number representing the current week.

And lastly the reward is:

$R(s, a) = x^h * PenaltyAtWeek(h)$

where x is the discount factor and h is the number representing the current week and $0 <= h <= N$ where N is the maximum number of weeks we are accounting for. PenaltyAtWeek(h) is another function which calculates the penalty based on a pre-defined cost and the number of failures for that week. PenaltyAtWeek(h) = c * NumOfFailures(h) where c is the pre-defined cost. NumOfFailures(h) is the number of failures for week-h. The number of failures is calculated based on the state s' that is a result of the action a taken in state s and the consumption of the user.

Conceptual Arguments

We defined our states and actions as tuples of the amount of each item because this definition is complete and easy to translate into code. Our transition function gives us a tuple of probabilities instead of one probability value because each item type has its own probability matrix and we need to take each one of this into account when calculating the new value for a state. Because of this, we can be sure that the new value for our state has the probabilities of each item factored into it.Our reward function is simply the total amount in the current state after the purchasing period minus the capacity of the fridge.

This is because we never really know if the user will eat a particular food item. What we know for sure is that as long as the fridge is full, whilst keeping into account the eating patterns of users, we will minimize the amount of failures. Therefore, the less items we purchase, the lesser our reward is. This ensures that the fridge will order as much as it can with respect to the users' eating pattern. Our method simply iterates over values of each state once to reduce the amount of time we take for offline computation. This allows us to generate a working policy quicker, which is useful when the method is used for large fridges. However, this means that we will, most likely, not generate the optimal policy. As a result, we will be failing more often,

Description of Method

Our method is a modification on Value Iteration, where instead of iterating multiple times to get the optimal value, we only iterate once, attempting to generate the policy closest to the optimal policy. We also generate all the possible states and actions for a given fridge. We do this to minimize the time taken for offline computation and simply get the policy during the actual simulation runs to save time. This assists us when tackling fridges larger than small size. Furthermore, this reduces the chances of going over the amount of time we have for offline computations since we know we are minimizing the amount of time we take to compute.

The Level Our Method Solves and Why

We believe that this method will solve scenarios involving tiny, small and medium sized fridges. This is because, although it doesn't generate the optimal policy, the method still generates a valid policy. Furthermore, using our reward function, we ensure that the fridge always buys as much as possible. If we happen to by the wrong items for one week, our previous purchases will be useful in future weeks.

This method may not solve all possible scenarios involving tiny, small and medium sized fridges simply because the policy generated is not optimal. Some scenarios may require the fridge to be nearly flawless, which will be nearly impossible to achieve with this method.

This method does not solve fridges above medium in a reasonable amount of time due to the increase in complexity for the state and action space. Large fridges can store 10 items (3 more than the medium fridge) and purchase 5 items (2 more than medium) and the Super Large fridges can store 4 times the amount the large fridges can store and can order 5 times the total amount the medium fridge can order. Generating all the possible states and actions for the large and super large fridge will definitely take more than 5 minutes of offline computation, as indicated by our experimental data below.

Experimental results

The following data are averages and totals based on 10 simulation runs for each input file shown.

Input file	results
tiny-v1	Number of weeks: 15 Cost: -10.0 Discount Factor: 0.975 Fridge: tiny Total Penalty: 1083.76595 Total max penalty: 3791.751773 Overall penalty ratio: 0.285823 - Solved in time
tiny3.txt	Number of weeks: 15 Cost: -10.0 Discount Factor: 0.9995 Fridge: tiny Total penalty: 836.866972 Total maximum penalty: 4484.284074 Overall penalty ratio: 0.186622 -Solved in time
small-v1.txt	Number of weeks: 15 Cost: -10.0 Discount Factor: 0.975 Fridge: small Total penalty: 1845.815835 Total maximum penalty: 6319.586289 Overall penalty ratio: 0.292079 - Solved in time
medium1.txt	Number of weeks: 15 Cost: -10.0 Discount Factor: 0.975 Fridge: medium Total penalty: 1386.998886 Total maximum penalty: 8847.420804 Overall penalty ratio: 0.156769 - Solved in time
large1.txt	- Unable to solve in time.