Project Report

CMPT 417

Shreya Jain (301391230)
Submitted To: David Mitchell

December 1, 2019

Pizza Coupon Problem

The problem chosen for this project is the pizza coupon problem. The description of the problem is: There is a list of pizzas that need to be bought, each with a price. There is also a collection of coupons that are of the form "buy x pizzas, get y pizzas free". Each of these coupons can be used to used to get a specified number of pizzas free, provided some other specified number of pizzas are paid for. Each pizza that is free by applying a coupon c must have a price no more than that of the cheapest paid-for pizza used to justify using coupon c. The objective is to find a selection of coupons, and "assignments" of pizzas to coupons (the paid pizzas and free pizzas associated with each coupon is used), that allows getting all the desired pizzas for total cost at most k. i.e. obtain all the ordered pizzas for the least possible cost. The actual cost is calculated by multiplying all the paid for pizzas by their price.

Using MiniZinc (the solver system used for the project), optimization of the problem was worked upon. Essentially, the total cost function of the pizzas was minimized to obtain the least possible cost to get all the desired pizzas. An assumption made here was that any pizza used to justify a coupon can be used only once.

Problem Specification

Let the instance vocabulary be [price, buy, free, n, m] where:

- n is number of pizzas and is a constant
- m is number of coupons and is a constant
- Unary function price: [n] → N, giving the price for each of the n pizzas
- Unary function buy: [m] → N, giving the number of paid pizzas required to justify using each of the m coupons;
- Unary function free: [m] → N, giving the number of free pizzas that can be obtained by using each of the m coupons;
- Cost bound k ∈ N

The following vocabulary symbols will also be further used:

- Unary relation symbol Paid, for the set of pizzas will be paid for
- Unary relation symbol Used, for the set of coupons that will be used
- Binary relation symbol Justifies, where Justifies(c, p) holds if pizza p is one of the pizzas we will pay for to justify using coupon c
- Binary relation symbol UsedFor, where UsedFor(c, p) holds if p is one of the pizzas we get free by using coupon c
- Φ is the formula for solving the problem

The constraints applicable are:

1. Pay for exactly the pizzas that are not obtained by applying the vouchers
2. Used is the set of applied vouchers
3. Any voucher that is applied must be Justified by sufficiently many purchased pizzas
4. The number of pizzas any voucher is applied to is not too large
5. Each free pizza costs at most as much as the cheapest pizza used to justify the application of the relevant voucher
6. Every pizza used to justify applying a voucher is paid for
7. The total cost is not too large (total of all pizzas)
8. Require that J(v,p) and A(v,p) hold only pairs consisting of a voucher and a pizza.
9. Any pizza used to justify a voucher can only be used once

The problem specification is then given by:

Given: A structure A for the vocabulary of the formula φ, and a valuation σ, $\#(\tilde{x}, \varphi(\bar{x}\,\bar{y}))A[\sigma]$ is the number of distinct tuples $\bar{a} \in A\ k$, where $k = |\bar{x}|$, such that $A \models \varphi[\sigma(\bar{x} \to \bar{a})]$.

To find: A structure B that is an expansion of A to the vocabulary [price, buy, free, n, m, Paid, Used, Justifies, UsedFor] and that satisfies the following formula φ:

$\Phi = (\forall p[\text{Paid}(p) \leftrightarrow \neg\exists c\ \text{UsedFor}(c, p)]) \wedge (\forall c[\text{Used}(c) \leftrightarrow\ \exists p\ \text{UsedFor}(c, p)]) \wedge$

$(\forall c[\text{Used}(c) \to \#(p, \text{Justifies}(c, p)) \geq \text{buy}(c)]) \wedge (\forall c[\#(p, \text{UsedFor}(c, p)) \leq \text{free}(v)]\ ) \wedge$

$(\forall c \forall p1 \forall p2[(\text{UsedFor}(c, p1) \wedge \text{Justifies}(c, p2)) \to \text{price}(p1) \leq \text{price}(p2)]) \wedge$

$(\forall p \forall c[\text{Justifies}(c, p) \to \text{Paid}(p)]) \wedge (\text{sum}(p, \text{Paid}(p), \text{price}(p)) \leq k) \wedge$

$(\forall c \forall p[\text{Justifies}(c, p) \to (c \in [m] \wedge p \in [n])]) \wedge (\forall c \forall p[\text{UsedFor}(c, p) \to (c \in [m] \wedge p \in [n])]) \wedge$

$(\forall c \forall p[\text{Justifies}(c, p) \to \neg\exists d\ (\text{Justifies}(d, p) \to (c\ != d))])$

NOTE: A complete code based on this specification is done in MiniZinc that can be found in the parent Zip file of this report.

Test Instances

A total of 10 test instances were constructed. 3 of them were taken from the official page of the LC/CP Programming Challenge to test the code with the given input and output samples. The remaining 7 of them were taken from the MiniZinc Challenge 2015 test instances. The test instances taken from there were larger in size and hence were chosen so as to check the accuracy of the code and whether it will work for comparatively larger values/data. While running the program with the test instances, default settings in the solver system were applied with only inputs differing every time.

Results

Input	Output	Run Time

n = 4;
price = [10,5,20,15];
m = 2;
buy = [1,2];
free = [1,1];
	cost(35)	218 msec

n = 4;
price = [10,15,20,15];
m = 7;
buy = [1,2,2,8,3,1,4];
free = [1,1,2,9,1,0,1];
	cost(35)	285 msec

n = 10;
price = [70,10,60,60,30,100,60,40,60,20];
m = 4;
buy = [1,2,1,1];
free = [1,1,1,0];
	cost(340)	411 msec

n = 20;
price = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10];
m = 5;
buy = [3,1,1,2,3];
free = [3,1,1,1,2];
	cost(120)	Stopped at 3min 11 sec

n = 18;
price = [243,7031,4313,8033,6321,1688,2132,4142,9593,5882,2167,8072,4972,4594,1139,6071,7764,500];
m = 4;
buy = [3,2,3,2];
	cost(29997)	5 min 42 sec

free = [9,5,9,5];		

n = 19;
price = [88,15,66,26,35,16,80,27,46,58,68,18,63,6,69,08,39,91,23];
m = 8;
buy = [8,8,7,8,7,7,7,8];
free = [7,7,0,7,0,0,0,7];
	cost(675)	Stopped at 9 min

n = 14;
price = [87,74,63,34,63,16,97,75,08,86,17,61,31,04];
m = 4;
buy = [7,9,9,1];
free = [6,5,3,7];
	cost(207)	229 msec

n = 10;
price = [50,60,90,70,80,100,20,30,40,10];
m = 9;
buy = [1,2,1,0,2,2,3,1,3];
free = [2,3,1,1,1,2,3,0,2];
	cost(210)	4 sec 315 msec

n = 20;
price = [95,96,96,61,29,79,71,18,36,53,52,64,71,49,71,84,31,15,18,58];
m = 7;
buy = [5,5,5,5,5,5,5];
free = [9,9,9,9,9,9,9];
	cost(607)	Stopped at 10 min

n = 20;
price = [70,10,60,60,30,100,60,40,60,20,80,70,90,56,90,87,50,50,20,80];
m = 8;
buy = [1,2,1,1,0,1,2,1];
free = [1,1,1,0,1,1,0,1];
	cost(747)	Stopped at 7 min