Automatic Recording Problem

In this programming assignment, you will use linear programming to obtain an upper bound on the objective for the integer automatic recording problem (ARP). Integer ARP is as follows: maximize user satisfaction with the contents of a recorder (e.g. Tivo) over a set period of time, given a listing of overlapping programs with varying durations and value to the user. The constraints are that only one program can be recorded at once, you must record a program in its entirety, and you have a finite recording capacity.

Input

As input, you are given the number of programs available for recording and the capacity of the recorder. You are also given, for each program, the start time, end time, duration, and associated profit.

Data

You will find the data in `/course/cs149/data/ARP_BENCHMARK/`. Sub-directories are named according to the parameters with which the test instances were generated. For a more in-depth description of the objective types, see the end of this document.

Output

You are to formulate your relaxation of the problem as a linear program. Once you have a formulation, you may either output the LP in a standardized format, such as `.LP`, to be fed into CPLEX, or you may use Concert / the CPLEX callable library to have formulation and optimization in a single executable.

Your Tasks

You are to write code which will formulate, and obtain the value of, a linear relaxation of an integer ARP defined by an input file. The quality of your relaxation is determined by how closely it matches the integer solution. Your goal should be to obtain as tight of a bound for the integer problem as possible. Please also hand in a description of your relaxation, and an explanation of why you believe it is a good upper bound.

Support Code

When you are ready to begin coding, copy the support code from `/course/cs149/asgn/arp/`. You can compile by typing:

```
cslab0a /course/cs149/asgn/arp % make
```

and run by typing:

```
cslab0a /course/cs149/asgn/arp % ./arp_bound <ARP instance file>
```

If you will be creating an `.LP` file, fill in the method `void arpInstance::writeLPFormulation(char* fname)`. If you will be using Concert or the CPLEX callable library, fill in the method `void arpInstance::solveLPFormulation()`. These methods, as provided, contain small examples of how to output an `.LP` file and how to use the ILOG Concert library.

CPLEX

The local reference documentation for CPLEX is located at `file:///com/cplex/doc/html/index.html`. If you choose to output your relaxation as an `.LP` file, you can obtain the relaxation value as follows:

```
pekoe /tmp % cplex
CPLEX> read tmp.lp
CPLEX> opt
```

Test Instance Parameters

To achieve scenarios of relevance for the real life application, each set of instances is generated by specifying the time horizon (720 or 1440 minutes) and the number of channels (20 or 50). The generator sequentially fills the channels by starting each new program one minute after the last. For each new program a class is chosen randomly. That class then determines the interval from which the length is chosen randomly. We consider 5 different classes. The lengths of programs in the classes vary from 5 ± 2 minutes to 150 ± 50 minutes. The disc space necessary to store each program equals its length, and the storage capacity is randomly chosen as 45% to 55% of the entire time horizon.

To achieve a complete instance, it remains to choose the associated profits of programs. We use four different strategies for the computation of an objective function:

- For the class usefulness (**CU**) instances, the associated profit values are determined with respect to the chosen class, where the associated profit values of a class can vary between 0 and 600 ± 200.
- In the time strongly correlated (**TSC**) instances, each 15 minute time interval is assigned a random value between 0 and 10. Then, the profit of a program is determined as the sum of all intervals with which the program has a non-empty intersection.
- For the time weakly correlated (**TWC**) instances, that value is perturbed by a noise of $\pm 20\%$.
- Finally, in the strongly correlated (**SC**) data files, the profit of a program simply equals its length.

The different objectives try to emulate some effects we believe to hold in real life instances. In the **CU** instances, for example, programs of the same class cause similar attractions. The **TC** and **TWC** instances cause many conflicts regarding the choice of programs that are being broadcast at the same time.