Project 3

Cam Hayes

The system works by initially creating a dictionary of attributes and their binary representation, encoding the attributes into a list composed of ordered lists for each object, a constraints set defined by the constraints file and the feasibility set generated by iterating over all of the encoded objects and checking them against the constraints. Because pySAT uses a binary integer representation for objects, the system maintains that structure in all its data objects and references a find\_in\_dict method when a textual representation is needed.   
  
After generating the workable data sets, the system engages the menu interfaces. Because of time/scope constraints, there is a functional method for penalty and qualitative logic for each of the main data processing components.

Penalty Logic:   
A penalty logic set is generated through the provided preference file. This creates a binary integer PySAT friendly interpretation of each line in the penalty logic file. The logic follows that a formula is split into clauses at AND, and their literals are evaluated at OR and NOT. An example output of a rule would be:

beef = -3, cake = 1; soup = -2

Rule: beef OR cake AND NOT soup

Output: [[-3, 1], [2]]

Processing penalty logic involves iterating over every rule for every feasible object. If a rule is solved by an object, then the penalty appended to the data structure for that object is 0. If it does violate the rule, the actual penalty is appended to the data structure for that object. Processing the penalty logic returns a list of processed object sets, where the 0th index is the object number and the remaining members are penalty costs incurred for rules n, n+1, … etc.

Exemplifying and optimizing penalty logic is a simple comparison of the sum of incurred penalties between each object.

Qualitative Choice Logic:

A similar logic set is generated for QCL, with the exception of a BT and IF delimiter. The object set for each QCL rule is a composite of the rules themselves, followed by the initial condition which is appended to the end of the list. Processing QCL occurs by first processing whether the initial condition is applicable. If so, each condition is processed until the first successful solution. If there is no solution or if the initial condition failed, the infinity value is appended to the object. The object set for processed QCL objects is formatted such that each member of the list if the preference found during evaluation. For optimization and exemplification of QCL sets, the program iterates over the vector components for two vectors, and determines whether or not comparison rules in Pareto optimization apply at each vector component. This is, for example, if x\_1 > y\_2, x\_2 < y\_2, x\_3 == y\_3, x\_4 OR y\_4 == INFINITY, etc. The result of that comparison is returned and evaluated by the calling method.

The remaining methods handle basic dispatch and utility services.

Initialization and Encoding Print:  
A screenshot of a computer

AI-generated content may be incorrect.

Feasibility Display:

A black screen with white text

AI-generated content may be incorrect.

Penalty Logic Table:  
A screenshot of a computer

AI-generated content may be incorrect.

Exemplification (penalty logic):  
A black screen with white text

AI-generated content may be incorrect.

Omni-optimization (penalty logic):

A black screen with white text

AI-generated content may be incorrect.

Qualitative Choice Table:  
A screen shot of a computer

AI-generated content may be incorrect.

Exemplification (Qualitative Choice):  
A black screen with white text

AI-generated content may be incorrect.

Omni Optimization (Qualitative Choice):

A black screen with white text

AI-generated content may be incorrect.