NAME

*.q – parameterized QUBOs

SYNOPSIS

*.q

DESCRIPTION

The **dw embed** command reads a text format input file with suffix .q that defines a parameterized Quadratic Unconstrained Binary Optimization (QUBO) problem. If successful, **dw embed** creates a binary format output file containing either a generated or user-specified embedding and a single parameterized quantum machine instruction (EPQMI). This describes the format of the .q file which is the input to **dw embed.**

QUBO files which can be parsed by **dw embed** have a very simple format. Blank lines are ignored. Comment lines must begin with a hash character. Apart from these, there are four other kinds of lines that may appear in a .q file, each of which may appear more than one time, and must appear in the following order:

param: defines a parameter to be provided to **dw bind**

var: defines 0/1 variable over which QUBO is minimized

term: defines a term appearing in the QUBO

assert:[*varname*:] defines an assertion

Both parameter and variable names are alphanumeric, case sensitive, must begin with an alphabetic character, and can contain underscores. Examples are: x, foo, ra3_Z.

Terms are simple arithmetic expressions involving constants, parameters and variables. Infix notation and parentheses are supported with operators *, /, +, -. Terms are of three kinds: constant, linear and quadratic. Constant terms do not contain any variables, linear terms depend linearly on a single variable, and quadratic terms are proportional to the product of exactly two distinct variables. Apart from variables, each term may include a single parameter. The objective associated with the .q file is obtained by summing all the term: lines in the QUBO.

Assert lines with a single colon do not include a parameter. Assert lines with two colons contain a parameter name between the two colons. In either case, an expression follows the last colon. The expression supports infix notation and parentheses and can contain any number of variables. The expression is intended to capture constraints that should be satisfied when the QUBO is minimized. These constraints are translated into the PQMI so that validation can be performed on samples drawn from a quantum machine instruction execution. If a parameter name appears, it associates the assertion with term(s) that are responsible for ensuring that the assertion is satisfied when the QUBO is minimized. An assertion is valid when it evaluates to 0 after values have been supplied for the variables appearing in it.

Here is a complete and syntactically correct .q file:

```
param: color
param: weight_r
param: weight_g
param: weight_b

var: red
var: green
var: blue

term: -1 * color * red
term: -1 * color * green
term: -1 * color * blue
term: 2 * color * red * green
term: 2 * color * red * blue
term: 2 * color * green * blue
term: weight_r * red
term: weight_g * green
term: weight_b * blue

assert:color: red + green + blue - 1
```

In the above example, the first six terms are chosen so that the QUBO is minimized when exactly one of the three variables has the value 1 and the other two have the value 0 (provided that the *color* parameter has a positive value). The assertion evaluates to 0 when this condition is true and to a non-zero value when this condition is false. The six terms all include the color parameter, and so the strength of the terms responsible for enforcing the assertion can be modified by adjusting *color's*

value. The final three terms in the .q file allow the three valid solutions to be weighted independently using additional parameters.

Here is the redisplayed QUBO as shown by **dw get qubo**:

```
**************** DW_qubo: PARAMETERS ****************
PARAMETER("color"): value=0.000000
PARAMETER("weight_r"): value=0.000000
PARAMETER("weight_g"): value=0.000000
PARAMETER("weight_b"): value=0.000000
**************** DW_qubo: VARIABLES ****************
VARIABLE("red"): id=0 value=0
VARIABLE("green"): id=1 value=0
VARIABLE("blue"): id=2 value=0
**************** DW_qubo: TERMS ****************
TERM(0): expr="(weight_r+((-1)*color)) * red"
TERM(1): expr="(weight_g+((-1)*color)) * green"
TERM(2): expr="(weight_b+((-1)*color)) * blue"
TERM(0,1): expr="(2*color) * red * green"
TERM(0,2): expr="(2*color) * red * blue"
TERM(1,2): expr="(2*color) * green * blue"
**************** DW_qubo: ASSERTS ****************
ASSERT(0): param="color" expr="(((red+green)+blue)-1)"
```

In the standardized QUBO format, there are sections for parameters, variables, terms and assertions. Variables are assigned numeric ids starting from 0. These numeric ids identify the variable(s) appearing in each term in the objective. Assertions are also numbered. Note that the input file contains two linear terms for each variable and these have been combined into a single term in the display.

During embedding, **dw embed** associates each variable with a chain of one or more physical qubits. If any of the chains have length greater than one, **dw embed** introduces an additional parameter whose default name is *param_chain* and which controls the strength of the interactions responsible for maintaining chain integrity. **Dw embed** uses sapi_findEmbedding() from the D-Wave C API to create qubit chains. The default parameters for this function are in effect when **dw embed** invokes the function. For example, the 1000 second default timeout means that if **dw embed** is unable to find an embedding, it will spend approximately 1000 seconds before exiting. If a valid embedding is found, the embedding

information is contained within the default.epqmi file and can be viewed using the **dw get embedding** command.

BUGS

Please report bugs to dwsupport@dwavesys.com.

COPYRIGHT

SEE ALSO

The output EPQMI file generated by **dw embed** is binary. To display it in readable format and to further manipulate it, use the dw(1) command.