List of Objective functions in HAS

Existing Optimization Models

Minimize an unweighted value

```
{
         "minimize":{
            "attribute":
               {
                  "distance_between":[
                     "customer_loc",
                     "vG"
                  ]
               }
         }
 }
```

Minimize a weighted value

```
{
    "minimize": {
        "attribute": {
            "product": [
                200,
                {
                    "distance_between": [
                        "customer_loc",
                        "vG"
                    ]
                }
            ]
        }
    }
}
```

Maximize an unweighted value

```
{
    "maximize": {
        "attribute": {
            "reliability": [
                "URLLC"
            ]
        }
    }
}
```

Maximize a weighted value

```
{
    "maximize": {
        "attribute": {
            "product": [
                200,
                {
                    "reliability": [
                        "URLLC"
                    ]
                }
            ]
        }
    }
}
```

Minimize the sum of unweighted values

```
{
    "minimize": {
        "sum": [
            {
                "distance_between": [
                    "customer_loc",
                    "vG"
                ]
            },
            {
                "distance_between": [
                    "customer_loc",
                    "vG"
                ]
            }
        ]
    }
}
```

Minimize the sum of weighted values

```
{
    "minimize": {
        "sum": [
            {
                "product": [
                    100,
                    {
                        "distance_between": [
                            "customer_loc",
                            "vG"
                        ]
                    }
                ]
            },
            {
                "product": [
                    200,
                    {
                        "hpa_score": [
                            "vG"
                        ]
                    }
                ]
            }
        ]
    }
}
```

New Optimization Model

Objective Function Object

Attribute	Required	Content	Values	Description
goal	Y	String	minimize, maximize	The goal of the optimization
operation_function	Y	Operation function Object		The operation function that has to be optimized

Operation function object

Attribute	Required	Content	Values	Description
operator	Y	String	sum, min, max	The operation which will be a part of the objective function
operands	Y	List of operand object	EIther an operation-function or a function	The operand on which the operation is to be performed. The operand can be an attribute or result of a function

operation-function operand object

Attribute	Required	Content	Values	Description
normalization	N	normalization object		Set of values used to normalize the operand
weight	N	Decimal	Default: 1.0	Weight of the function
operation_function	N	operation function object		

function operand object

Attribute	Required	Content	Values	Description
normalization	N	normalization object		Set of values used to normalize the operand
weight	N	Decimal	Default: 1.0	Weight of the function
function	N	String	distance_between, latency_between, attribute	Function to be performed on the parameters

fucntion_params	N	dict		parameters on which the function will be applied. The parameters will change for each function.

Normalization object

Attribute	Required	Content	Values	Description
start	Y	Decimal		Start of the range
end	Y	Decimal		End of the range

JSON Schema

Examples

1. Minimize an attribute of the demand

```
{
    "goal": "minimize",
    "operation_function": {
        "operands": [
            {
                "function": "attribute",
                "params": {
                    "attribute": "latency",
                    "demand": "urllc_core"
                }
            }
        ],
        "operator": "sum"
    }
}
```

2. Minimize the sum of the distance between the demand and the customer location.

objective function - distance_between(demand, location) + distance_between(demand, location)

```
{
   "goal": "minimize",
   "operation_function": {
      "operator": "sum",
      "operands": [
         {
            "function": "distance_between",
            "weight": 1.0,
            "params": {
               "demand": "vG",
               "location": "customer_loc"
            }
         },
         {
               "function": "distance_between",
            "weight": 1.0,
            "params": {
               "demand": "vFW",
               "location": "customer_loc"
            }
         }
      ]
   }
}
```

Scenario:

Minimize the sum of latencies of slice subnets

objective function - latency(demand) + latency(demand)

```
{
   "goal": "minimize",
   "operation_function": {
   "operator": "sum",
   "operands": [
      {
         "function": "attribute",
         "weight": 1.0,
         "params": {
            "demand": "urllc_core",
            "attribute": "latency"
         }
      },
      {
         "function": "attribute",
         "weight": 1.0,
         "params": {
            "demand": "urllc_ran",
            "attribute": "latency"
         }
      }
   ]
 }
}
```

Scenario:

Max [sum (W_bw * min (ran_nssi_bw, core_nssi_bw, tr_nssi_bw), 1/(W_lat * (sum (w1 * ran_nssi_lat, w2 core_lat, W3* tn_lat)))]

```
{
   "goal": "maximize",
   "operation_function": {
   "operator": "sum",
   "operands": [
```

```
{
   "operation_function": {
       "operator": "min",
       "operands": [
           {
                "weight": 1.0,
                "function": "attribute",
                "params": {
                   "demand": "urllc_core",
                   "attribute": "throughput"
                }
           },
           {
                "weight": 1.0,
                "function": "attribute",
                "params": {
                   "demand": "urllc_ran",
                   "attribute": "throughput"
                }
           },
           {
                "weight": 1.0,
                "function": "attribute",
                "params": {
                   "demand": "urllc_transport",
                   "attribute": "throughput"
                }
           }
       ]
   },
   "normalization": {
      "start": 100,
      "end": 1000
   },
   "weight": 2.0
},
{
   "operation_function": {
       "operator": "sum",
       "operands": [
           {
               "weight": 1.0,
               "function": "attribute",
               "params": {
                   "demand": "urllc_core",
                   "attribute": "latency"
                }
           },
           {
               "weight": 1.0,
               "function": "attribute",
               "params": {
                   "demand": "urllc_ran",
                   "attribute": "latency"
                }
           },
           {
               "weight": 1.0,
               "function": "attribute",
               "params": {
                   "demand": "urllc_transport",
                   "attribute": "latency"
                }
           }
       ]
   },
   "normalization": {
      "start": 50,
      "end": 5
   },
   "weight": 1.0
```

```
      }
    ]
  }
}
```

normalization:

function(value, range(start, end), weight)

All ranges are converted to 0 to 1. The inverse operation is not needed since it is already implied in the range.

normalized value = (value - start) / (end-start)

Eg:

latency range: 50 ms to 5 ms

candidate latency	Normalized value
20 ms	0.667
40 ms	0.222

throughput range: 100 Mbps to 1000Mbps

candidate throughput	Normalized value
300 Mbps	0.222
800 Mbps	0.778