Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

GADGET: A Green Assessment and Decision GuidancE Tool for Optimal Investment in Inter-dependant Energy Infrastructures

Bedor Alyahya
balyahya@gmu.edu

Alexander Brodsky
brodsky@gmu.edu

Technical Report GMU-CS-TR-2023-3

Abstract

In response to the urgent need for climate action and the transition to cleaner energy, organizations and businesses are setting greenhouse gas emission reduction targets to achieve carbon neutrality. Achieving these targets requires informed investment decisions in a complex mix of interconnected infrastructures and resources related to energy service modalities.

This paper introduces GADGET, a Green Assessment and Decision Guidance Tool. GADGET assists stakeholders in optimizing investment choices in various energy service modalities. It takes into account interconnected infrastructures, greenhouse gas emission reduction, total cost of ownership, and can be extended to cover additional performance metrics. GADGET's uniqueness lies in its ability to generate machine-generated mathematical programming (MP) models through a library of power-centric analytical models. Additionally, it is equipped with a pre-processor engine that generates representative operational demand windows, such as daily patterns. This feature reduces the number of decision variables while accounting for operational interactions. Empirical studies confirm GADGET's applicability to large-scale investment challenges, ensuring effective and efficient decision-making in energy systems.

1 Introduction

In response to the climate crisis and the need for transition to cleaner energy, many institutions and businesses make commitments to climate action by setting greenhouse gas emission reduction targets to reach carbon neutrality. To achieve these targets, stakeholders need to decide on how to invest in a mix of heterogeneous interrelated modalities - infrastructures and resources - including (1) building energy efficiency and electrification; (2) heating and cooling efficiency at central plant and buildings; (3) local and contracted renewable sources of energy and co-generation; (4) energy storage; (5) management of schedulable loads such as for HVAC and EV charging; (6) peak demand management vis-à-vis utility contracts; and (7) renewable energy certificates and carbon offset credits. Thus, there is an urgent need for a decision guidance tool to help stakeholders with actionable recommendations on a mix of the inter-related infrastructures and resources. Such a tool must enable stakeholders to make informed Pareto-optimal trade-offs between competing objectives and performance indicators such as (1) GHG emission reduction, (2) total cost of ownership over a time horizon and implicit price of GHG reduction, and (3) availability, vulnerability and resilience. Furthermore, the tool must support solving a variety of investment problems across different energy system architectures without the need to solve each specific problem from scratch. This will enable building rapid solutions that are cost-effective, yet flexible and with a high level of accuracy.

However, crafting clear practical recommendations on infrastructure investments is inherently complex due to (1) the complexity of the heterogeneous infrastructures as their components operationally interact with each other, e.g., solar generation, energy storage and peak load reduction; (2) consideration of trade-offs between financial, environmental and quality-of-service performance indicators; (3) investment return dependency on efficiency of operation, which is transient due to stochastic nature of supply and demand; and (4) rapid changes in infrastructure technologies. Because of these challenges and despite remarkable improvements in the energy efficiency and individual power solutions, investments in heterogeneous infrastructures often fail to realize their full potential, resulting in significant waste of resources, which could be avoided.

A substantial body of research has been devoted to the prioritization and optimization of investments in energy systems, as highlighted by [1]. One prominent direction

within this domain has emphasized the development of models specifically designed to address challenges inherent to the energy sector. These models are curated with specific objectives, whether it's to align with targeted performance metrics, as seen in [2, 3, 4], address greenhouse gas emission mitigation [5, 6], refine system designs [7, 8], or focus on niche power systems, such as hybrid power desalination systems [9]. They aim to resolve particular investment challenges in various infrastructure segments, including power production [10, 11], distribution [5], and transmission [12]. However, these approaches do not consider or leverage the interconnected nature of diverse infrastructures. This oversight can lead to missed cross-silo investment opportunities and result in sub-optimal results.

Another line of research, exemplified by works [13] and [14], focuses on studying the inter-dependencies across a wide spectrum of infrastructures. While this broader perspective is essential, these studies are limited to financial aspects, but do not model detailed engineering and operational interactions of of such infrastructures. Relying on this financial-centric perspective puts the burden on users to come up with financial data they may not visibility into and thus result in superficial or unrealistic assumptions and results.

A promising approach is taken in HOMER [15], a tool for modeling and analyzing a microgrid. HOMER allows to model a flexible micro-grid composed of various components, and simulate its operation over a time period. HOMER also has a layer to support simulation-based black-box optimization. However, it does not support mathematical programming based optimization, such as methods for Mixed Integer Linear Programming (MILP) models, which are sufficient for GADGET type problems. Whereas, for problems amenable to mathematical programming, MP solvers are typically superior to simulation-based methods in terms of optimality of results and computational efficiency. Most importantly, HOMER was not designed for multi-period investment optimization under the assumption of optimal operational controls, which is the focus of this paper.

To overcome these limitations, we introduced the Service Network Investment Model (SNIM) and a decision guidance system in [16] based on mathematical programming. This system offers actionable, Pareto-optimal investment recommendations for heterogeneous infrastructure service networks under the assumption of optimal operational controls over the time horizon. However, [16] did not specifically address achieving carbon neutrality, nor did it focus on the investment modalities in electric power microgrids, such as renewable energy, energy storage, and utility contracts, and the associated models that support these modalities. Furthermore, SNIM leaves open the problem of generating its input data that is not readily available. Most critically, the problem of analysis and pre-processing of long term power, heating, cooling and gas demand to generate a manageable size set of typical operation demand windows (e.g., days) was left open.

To bridge this gap, the core contribution of this paper is the development of GADGET - a Green Assessment and Decision GuidancE Tool to make Pareto-optimal actionable recommendations on a mix of investment modalities including solar panels, energy storage, transformers, energy contracts, gas and electric boilers, renewable energy certificates (RECs) and carbon offsets. The GADGET model borrows from SNIM [16] and formalizes cash flows and performance indicators based on two key dimensions: (1) investment controls set for a given investment time horizon (e.g., 20 years), often segmented into multiple investment periods (e.g., every 2 years), and (2) operational controls specifically geared towards shorter operational windows, such as days, further divided into operational intervals (e.g., every 30 minutes), which are often controlled using an energy management systems (EMS.) Such granularity and precision in modeling are essential to fully capture the operational interactions that occur in interconnected infrastructures such as smart grids, renewable energy sources, energy storage systems, schedulable loads. This is especially true when these interactions occur over short operational intervals. To manage the vast number of operational decision variables, the model segments the investment periods into a sequence of Representative operational windows. These windows encapsulate consistent operational patterns, distinguishing, for instance, between weekend and weekday modes of operation. A distinguishing feature of GADGET is its ability to produce machine-generated mathematical programming (MP) models, using the system architecture proposed in [17, 18] and decision-guidance analytics language proposed in [18].

As part of GADGET development, we first have significantly enhanced the SNIM model with the model layer of financial instruments, such as RECs and carbon offsets, which must be handled differently than equipment components with operational interactions. Second, we modeled a range of micro-grid components including solar panels, energy storage devices, transformers, energy contracts, gas and electric boilers. These models formalize both operational interaction with the micro-grid service network, as well as computing financial metrics over long time horizons. Third, and most critically, we developed a pre-processor engine, based on a range of clustering algorithms, to generate representative daily operational demand windows. This allowed us to represent the demand pattern over each investment period as a sequence of representative operational windows, thereby significantly reducing the number of binary decision variables for scaling up optimization. Finally, we conducted two empirical studies: (1) to prove the feasibility of GADGET application to large-scale investment problems and (2) to assess the accuracy of GADGET's pre-processing algorithm by comparing computed KPIs

using its generated operational windows against actual supply and demand patterns.

This paper is organized as follows. Section 2 provides a comprehensive review of the Service Network Investment Model (SNIM). In Section 2.1, we delve into the architecture of GADGET. Section 2.2 describes the structure of the time horizon within which the model operates. Section 2.3 emphasizes the GADGET component: the pre-processor engine. Section 3 is dedicated to the formalization of the Service Network Investment Model (SNIM). The results of our preliminary experimental study are discussed in Section 5. Finally, Section 6 revisits the main points of our research and offers concluding remarks.

2 Service Network Investment Model: High-Level Overview

The Service Network Investment Model (SNIM) computes during each operational interval within the time horizon two essential components: (1) a diverse set of metrics covering financial, operational, environmental, and quality factors, and (2) constraints related to capacity, supply, demand, and other business considerations. These metrics and constraints are influenced by a mix of fixed and adjustable parameters. The model employs a bottom-up approach, enabling constraints and metrics at the operational level (often occurring at intervals as short as 30 minutes) to capture the interaction of the infrastructure. This interaction, in turn, influences these metrics and constraints at a higher level. Ultimately, this guidance extends to the controlled parameters designated for investments in specific modalities.

To comprehend how SNIM operates, we start by explaining the input structure, which involves a central component encompassing the hierarchical arrangement of all components within the power micro-grid, known as the "Service Network." In Figure 1, we use a campus power Service Network as an example to illustrate the hierarchical structure of interconnected services, represented by nested boxes. These services include various components such as cooling and heating plants, renewable energy sources, demand points (depicted as buildings), contractual agreements, financial instruments, and more. Together, they depict the flow of resources (e.g., power, fuel,...), and the state (e.g., battery charging level) at any given time within the network. At the base the hierarchy, we find what we call 'atomic services,' depicted as dotted boxes (e.g., batteries, local generator,...). Whether these instances of atomic services are currently owned or under consideration for potential investment, each comes with associated fixed and controllable parameters. These parameters define how every infrastructure is supposed to operate. For example, in the case of a battery, we can have parameters such as battery efficiency, charging and discharging rates, the invested number of units, and the required capacity per unit, among others, as illustrated in Figure 2. The model then collaborates with the extendable library, which houses all the analytical models of different types (e.g., the battery analytical model), to generate the metrics and constraints at the atomic level. Each atomic analytical model within the library provides a mathematical representation at the operational level to balance and compute the inflow and outflow of resources, and at a more comprehensive level spanning the investment time horizon to calculate the metrics. During the execution of the SNIM, the system systematically consolidates flows, metrics, and constraints, starting from the base level and moving upwards for each operational interval (e.g., every 30 minutes) within each window throughout the specified time horizon. Following this consolidation process, the system enters an iterative phase, revisiting the entire time horizon to generate metrics and constraints. This iterative step leverages the knowledge acquired about flows across the complete time horizon, resulting in a comprehensive set of metrics and constraints that encapsulate the intricacies of the system's behavior over time. This aggregated metrics and constraints are propagated to the *Financial Instruments* layer, initiating an iterative process over the financial instrument analytical models (e.g., RECs, Offset). This iteration aims to generate the metrics associated with these instruments and subsequently adjust the cash flow within the Service Network. From there, the SNIM can be utilized within the GADGET tool to solve the investment problem, which will be explained in the next section. A couple of notable features about the model include its capability to work with the extensible library of modalities. This ensures that new component models can be integrated without modifying the existing SN model or previously defined components. Examples of potential augmentations to the component models might include installing Uninterruptible Power Supply (UPS) units for redundancy, incorporating demand switch units for shifting from gas to electricity, or implementing elimination strategies using carbon capture technology. These enhancements can span various facets, encompassing power generation, transmission, and distribution to renewable energy sources and advanced energy storage solutions. The formal description of the extended SNIM model is given in Section 4.

2.1 GADGET Architecture

Figure 3 depicts GADGET high-level architecture. GADGET comprises five main layers: the Graphical User Interface (GUI), Pareto Optimal database, Graph Generator, Decision Guidance Management System (DGMS), Pre-processing Engine, and External Tools. The GUI provides users with the capability to access summaries of optimized problems, visualize diverse charts, compare scenarios, explore specific solutions, and conduct 'what

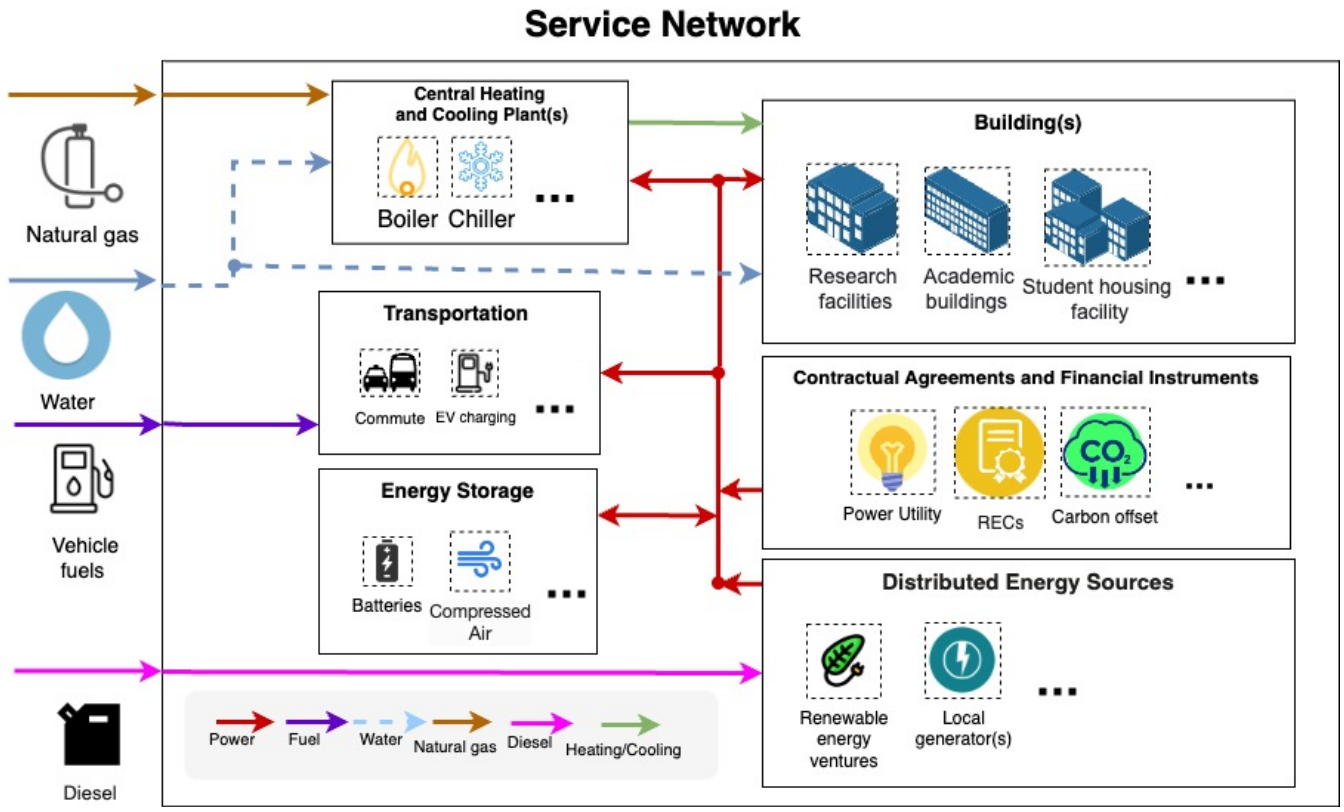

Figure 1: Service Network

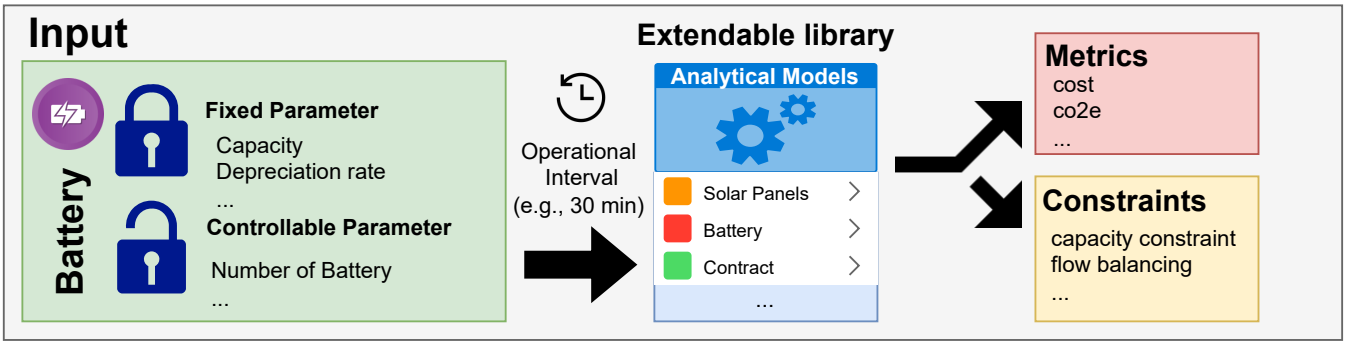

Figure 2: Atomic Analytical Models

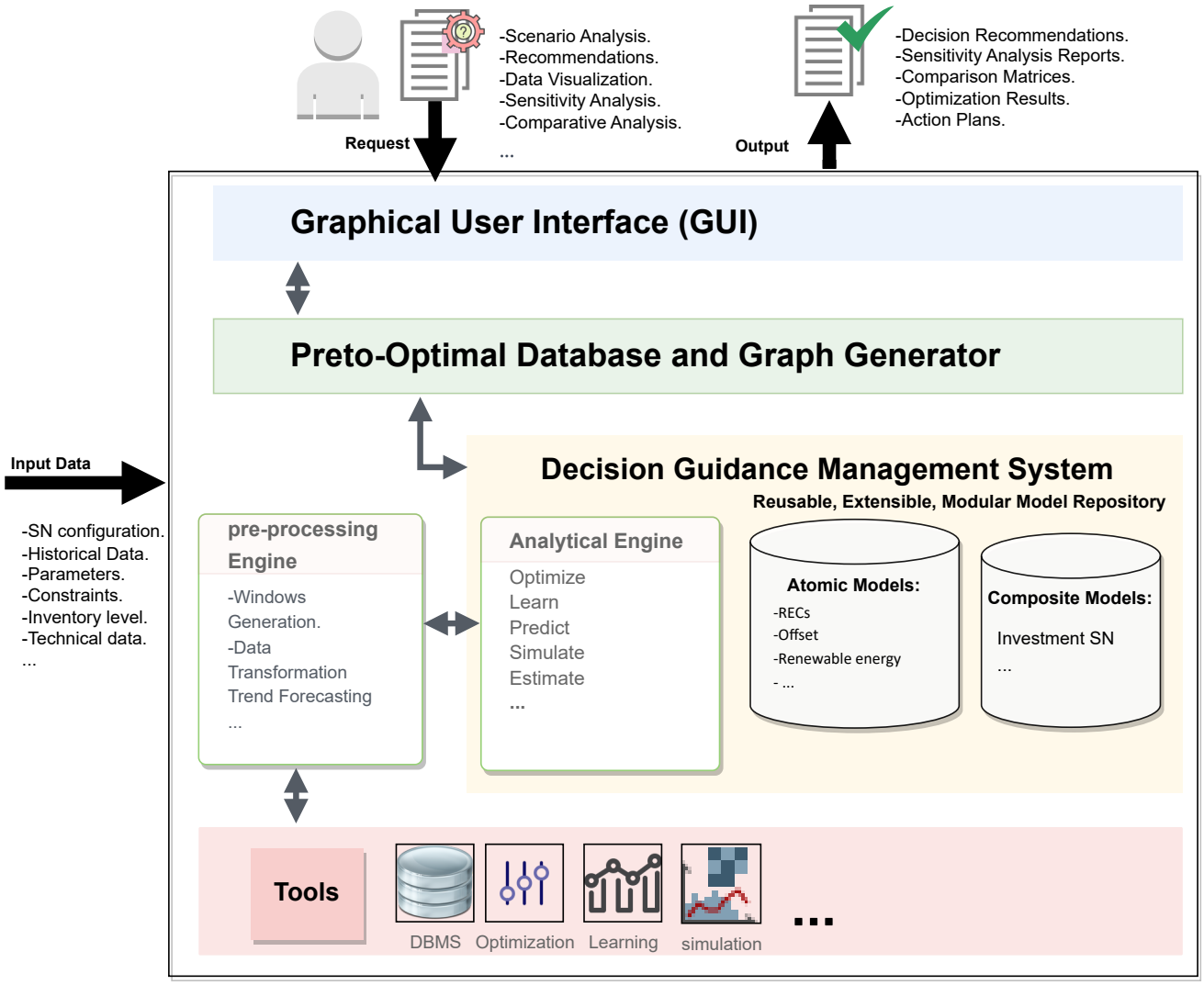

Figure 3: GADGET Architecture

if' and sensitivity analyses on selected scenarios.

The Pareto Optimal database and Graph Generator respond to user requests and work closely with the DGMS to undertake tasks necessary to fulfill the requests, such as optimization, prediction, and learning. These tasks are steered by the analytical engine within the DGMS, which also maintains a library of reusable, modular, and composable models. The library currently includes models for various modalities, including power storage solutions, solar panels, gas and electric boilers, carbon offset credits, Renewable Energy Certificates (RECs), transformers, and power utility contracts. The DGMS architecture is intentionally designed for effortless integration with various external tools, as depicted in the lower tier of the system. This integration empowers the DGMS to seamlessly collaborate with external tools such as CPLEX for optimization, without necessitating a direct binding to any specific tool. This flexibility is achieved through a translation process facilitated by the use of the Decision Guidance Analytical Language (DGAL), which was introduced in [19].

To generate graphs that require multiple optimization runs, such as comparison graphs, the Graph Generator can draw upon multiple pre-stored optimizations from the Pareto Optimal database. If a specific optimization scenario isn't already stored, the Graph Generator employs the analytical engine. This engine requires several inputs, including the SN configuration that blends time-centric (e.g., time horizon setting) and financial parameters (e.g., discount rate), the selection of models from the library, and their fixed parameter settings, leaving the rest of the parameters to be optimized. Additionally, for tasks like data transformation, trend forecasting, and notably, window generation, the Graph Generator relies on the Pre-processing Engine. During the window generation process, this engine adjusts the supply and demand trajectory to align with a given time horizon. A more detailed discussion on the structure of the time horizon and the window generation process will follow in the upcoming sections.

2.2 Time Horizon Segmentation

The time horizon (e.g., 25 years) is segmented into a sequence of investment periods possibly different length (e.g, 3,3,4,5,10), assuming that investments can take place at the commencement of each period, ensuring the availability of infrastructure from the outset. Financial aspects, including payments for both investments and operations, are specified using a more refined financial sequence (e.g., daily), as illustrated in the cash flow sequence in Figure 5. Utilizing this sequence, the model gains the capability to articulate multiple payments for each infrastructure investment, as well as the definition of multi-cycle payments for operational expenses (e.g., bills, payroll, etc.) Regarding the operational aspects of the system, each investment period is defined by a sequence of operational windows, wherein each window is composed of smaller, refined intervals that illustrate the system's interactions at the operational level (e.g., every 30 minutes). The following section provides a detailed description of how we create these operational windows.

2.3 Pre-processing: Generation of Representative Operational Windows

Representing all individual windows over the entire extended horizon creates a major optimization scalability problem. Each operational interval (e.g., 30 minutes) over investment periods over the time horizon (e.g., of 20 years) will have binary on/off decision variables. This does not scale, according to our experimentation, to real size problems when using MILP solver (CPLEX.) To address this, and to harness the operational interactions of interconnected infrastructures (e.g., renewable energy, power storage, and schedulable loads) over shorter operational intervals (e.g., 30 minutes), the engine creates representative windows as proxies for actual windows that exhibit similar supply and demand patterns.

To generate these representative windows, a systematic process is employed to preserve critical factors such as peak demand and total demand. Initially, within each investment period (e.g., 3 years), the engine categorizes all windows based on their power peak demand. This categorization employs a clustering (bucketing) technique, where windows with peak demand within a small distance (ϵ) from the maximum peak are grouped into a single category. As the process extends away from the overall maximum peak observed across all windows, the value of (ϵ) increases for each subsequent category. This ensures that, as peaks decrease in magnitude, a broader range of windows is accommodated within the same category. This step is essential to accurately represent windows with elevated peaks, a crucial consideration due to their direct impact on demand contract charges (refer to Figure 4, step 2).

Subsequently, the engine refines the categorized buckets using the Mean Shift clustering algorithm to further segment them based on additional dimensions, such as the normalized daily accumulated solar radiation and normalized daily heat demand (see Figure 4, step 3). The bandwidth, which determines the sensitivity to fine details in the data, is adjustable based on the user's preference for refining these windows. For each refined category, the engine creates a representative window that closely aligns with the aggregate metrics of the windows in that particular category. The criteria for this are:

1. The aggregated power and heat demand, along with other supply and demand dimensions like solar radiation from all windows in a category, should match that of the representative window when multiplied by the number of windows in that category.

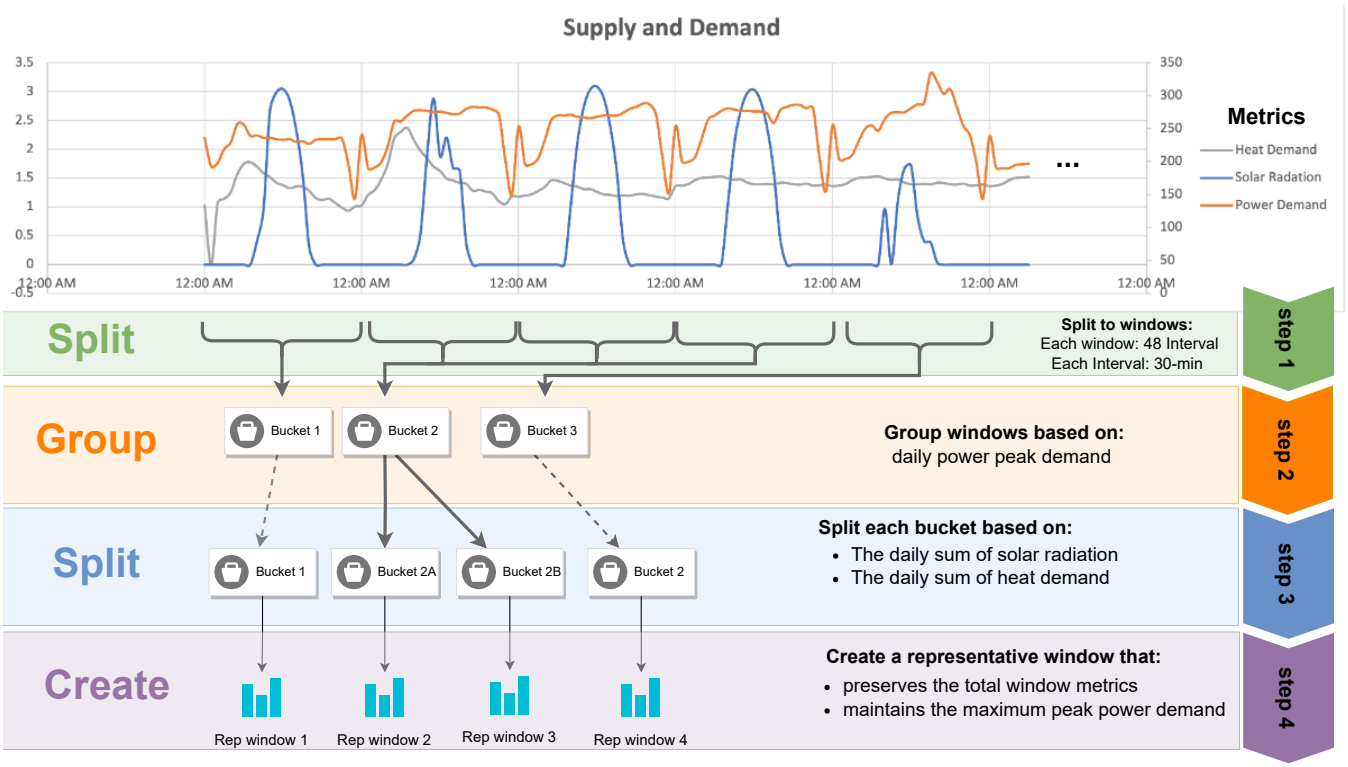

Figure 4: Windows Generation

2. The peak power demand of the representative window should mirror the highest among all windows in the designated category (See Figure 4, step 4).

It's important to note that there are two bucketing phases. The primary phase underscores the significance of peak heat demand, while the subsequent phase pivots towards other clustering dimensions.

3 Service Network Investment Model: Formalization

3.1 Optimization Problem

The optimization problem is shaped by the output from the analytic performance model (AM). This model computes various performance metrics, such as cost and carbon emissions. Additionally, it integrates feasibility constraints, reflecting both constant and variable operational and investment parameters across the entire investment time horizon. For a formal depiction, let us define the analytical performance model, AM, as follows:

$$AM : IN \rightarrow OUT \tag{1}$$

Here, AM forms a valid output instance $out \in OUT$, representing performance metrics like cost or emission. This is derived from a valid input $in \in IN$ which contains both fixed and controlled operational and investment parameters.

Given the aforementioned context, we can express the investment optimization challenge as:

$$\begin{aligned} \min_{in \in IN} \quad & obj(AM(in)) \\ \text{s.t.} \quad & C(AM(in)) \end{aligned} \tag{2}$$

In this formulation, Obj is an objective function that quantifies the objective's real value in $\mathbb{R}$ using a valid output instance $out \in OUT$. Meanwhile, C functions as a constraint evaluator. It returns either *True* or *False*, indicating whether the solution is feasible and meets all constraints or not.

Representing the optimization problem as an outcome of the Analytical Model (AM) rather than expressing it directly is advantageous because it can effectively address a wide range of investment optimization challenges. This approach offers flexibility in accommodating various infrastructure configurations, objectives, and timeframes without the need to modify the AM.

In what follows, we'll introduce a notation to represent a set of *key-value* pairs:

$$\mathtt{m} = \{key_1 : value_1 , key_2 : value_2 , \quad \ldots , key_n : value_n\}$$

The *keys* serve as distinct identifiers. Notably, this collec-

tion signifies a relation

$$m : \{key_1, \ldots, key_n\} \rightarrow \bigcup_{i=1}^{n} D_i$$

from the set of keys $\{key_1, \ldots, key_n\}$ to the union of the domains $\bigcup_{i=1}^{n} D_i$, with D_i being the domain of values linked to key_i. Hence,

$$m(key_i) \in D_{i \in \{1 \ldots n\}}$$

For referencing all keys related to the set m of key-value pairs, we employ the expression $keys(m) = \{key_1, \ldots, key_n\}$. Moreover, we introduce a list notation in the form $l = [a_1, \ldots, a_n]$. This is interpreted as a function

$$l : \{1, \ldots, n\} \rightarrow D$$

where D signifies the domain of list values. Here, $l(i) = a_i$ denotes the i-th list element, and the expression $length(l) = |l|$ returns the list's element count.

Using the notations mentioned above, we are equipped to elaborate on all previously discussed components. We initiate with a valid model output instance, *out*, showcased in section 3.2, proceed with the input instance *in* in section 3.3, and conclude by illustrating the analytical model which derives an output instance from the provided input instance.

With the notations explained, we'll first discuss the model output *out* in section 3.2, then the input *in* in section 3.3, and finally, how the analytic model processes the input to produce an output.

3.2 Service Network Instance: The Model Output

A valid *SN* output instance *out* comprises of a set of *key:value* pairs, structured as follows:

```
{config :<set of configurations>,
 rootServiceID :<root service ID>,
 instruments:<set of instruments>,
 services :<set of services>},
 adjustments: <value as in form 4>
```

Form 1: out

where *config* value ,extracted from the input, consists of *key:value* pairs structured as:

```
{financial:{financialInterval:<value>,
            intRate:<value>,...},
 intRatio:<value>,
 horizon: [p1,p2,...]}
```

Form 2: config value

In this context, *intRate* signifies the interest rate (e.g., 0.0001%) for each *financialInterval* (e.g., day), while *intRatio* (e.g., 24) indicates the number of operation intervals (e.g., of 1 hour each) within the financial interval (e.g., 1 day). The underlying assumption is that the time is split into operational intervals during which operations, such as machinery control, are conducted.

The *horizon* describes a list $[p_1, p_2, \ldots]$ of investment periods, each structured as:

```
{windows:{w1:{length:l1}, w2:{length:l2},...},
 winSeq:<sequence of window IDs>}
```

Form 3: period value

Here, each *period* represents a set of windows (e.g., summer day, winter day, etc.) that can have different lengths (e.g., day, week, etc.).

The *winSeq* denotes the sequence in which these windows appear within a particular investment period of the time horizon. As such, window IDs can recur to depict recurring window patterns.

In Form 1, the *instruments* comprise of a set of *key:value* pairs. Each *key* serves as a unique identifier for an instrument (like RECs), while the *value* provides details on the parameter and metrics associated with that specific instrument. Here's a breakdown:

```
 {metrics:{NPV:<list of values as in form 6>,
           cashFlow:<list of values as in form 6>,
           m1:<value as in form 5>,
           m2:<value as in form 5>,...},
  constraints:True or False}
```

Form 4: metrics and constraints

In this context, the value of each *metrics* encapsulates an array of additive metrics, denoted as $m_1, m_2, \ldots$. These metrics could represent various factors like cost or emissions. Furthermore, every metric value provides a breakdown of quantities specific to each investment period. As a result, the structure for each metric value is as follows:

```
{perPeriod:[v1,v2,...],total:<numerical value> }
```

Form 5: metric value

The *metrics* section includes a distinct *key* referred to as *cashFlow*, along with other financial metrics that are contingent on the *cashFlow*, such as the *NPV*.

The value of *cashFlow* is an array comprising various *payment* entries. Each of these payments adopts the format:

```
{interval: i,amount: <numerical value>}
```

Form 6: payment value

Here, every interval i designates a unique payment interval spanning the entire time horizon and is included no more than once in the list.

Moreover, the constraint values, as elaborated in Form 4, indicate the extent to which the instrument adheres to its specified constraints.

The *rootServiceID*, in Form 1, denotes the root service's ID, such as a micro-grid system. Specifics about the service network's layout and each service's parameters can be found within the *services* value. Here, services are identified either as *composite* or *atomic*. A *composite* service, like *Energy Storage*, encompasses one or more sub-services, and the IDs for these sub-services are listed under *subService*. The format for each *composite* service is illustrated below:

```
{type:"composite",
 inFlow:{f1:[v1,v2,...,vP],f2:[v1,v2,...,,vP],...},
 outFlow:{f1:[v1,v2,...,vP],f2:[v1,v2,...,,vP],...},
 metrics:<value as in form 4>,
```

```
constraints:True or False,
subServices:<set of service IDs>}
```

Form 7: composite service

In this structure, *inFlow* and *outFlow* have a set of flows {f_1,f_2,...} symbolizing the IDs of flows that enter and exit each service, respectively. Each flow's corresponding value is a list of period flows $[v_1, ..., v_p]$, each of the form:

```
{ w1:{qty: [q1,q2,...],total:<value>},
  w2:{qty: [q1,q2,...],total:<value>},...}
```

Form 8: period flow value

The *qty* list, represented as $[q_1, q_2, ...]$, indicates the flow quantity for each operational interval within specific windows, denoted by $\{w_1, w_2, ...\}$, for an investment period p. Conversely, *total* provides a summary of the entire flow spanning a single window during that period.

The *atomic* service largely mirrors the structure outlined in Form 7. However, there are some key differences. Firstly, it doesn't contain the *subServices* key-value pair. Secondly, its *type* references an *atomic* analytical model found within the library. Finally, this structure introduces some other unique key-value pairs that set it apart:

```
numUnitInvested:[v1,...,vp],
avaliable:[v1,...,vp],
initAvaliable:<value>,
capacityPerUnit:<value>,
numUnitON:<value as in form 10>,
state:<value as in form 10>,
lifeExp:<value>,
initLifeExp:<value>
```

Form 9: additional pairs

In this structure, the *numUnitInvested* and *available* values represent the counts of invested and available units for a specified service across various periods. The *initAvailable* designates the initial count of available units at the onset of the first period. Concurrently, *capacityPerUnit* provides the peak capacity each unit can accommodate. Moreover, the *numUnitON* denotes the count of units actively running (ON) during each interval of a window within the specified investment period. This leads to the following structured representation for *numUnitON*:

```
[{w1:[v1,v2,...],w2:[v1,v2,...],...},
  {w1:[v1,v2,...],w2:[v1,v2,...],...},...]
```

Form 10: operational value

The *state* is an optional set of parameters (e.g., accumulated kWh, battery charge level, etc.) used to track the service state. Each parameter represents a *key:value* pair, where the key signifies the ID of the parameter, and the value is in the form of Form 10, reflecting the service's state during each operational interval within the period's windows. The *lifeExp* represents the number of financial intervals during which the service is expected to function normally before requiring replacement when invested in. Conversely, *initLifeExp* indicates the number of financial intervals remaining for a service that is already in the inventory (i.e., not newly acquired.)

As for Form 1, the *adjustments* value aggregates the metrics of the *rootService* and the *instruments*. Moreover, the constraints here must satisfy those of the *rootService* and *instruments*, as outlined in Form 4.

In the following section, we'll outline the required input model used to create the output instance.

3.3 Service Network Instance: The Model input

The input model (*in*) mirrors the structure of the output with several distinctions: (1) The *metrics* and *constraints key:value* pairs are absent since they will be calculated from this input mode; (2) In the *composite* service, rather than presenting a list that details flow quantities, we substitute them with their lower bounds (*LB*); (3) Contrary to providing a list that illustrates the *state* in the *atomic* service, it's replaced by a singular value representing the *state* at the outset of the initial operational window; (4) For an *atomic* service and *instruments*, there's an added set of key-value pairs:

```
payments:{invPayments:<value>,
          opPayments:<value>}
typeSpecific:{<set of key:value pairs>}
```

Form 11: additional atomic key:value pairs

where the *invPayments* $\langle value \rangle$ represents a list of payments for investing in this atomic service during a particular period $p \in \{1, .., P\}$, formulated as:

```
{1:{{due:<value>,amt:<value>},
    {due:<value>,amt:<value>},...},
 ...,
 P:{{due:<value>,amt:<value>},
    {due:<value>,amt:<value>},...}}
```

Form 12: invPayment

Each *due* specifies the intervals (relative to the start of period p) when the amount (*amt*) is due, assuming the investment is made during period p. On the other hand, *opPayments* represents a sequence of operational payments, each associated with specific billing intervals. Thus, the *opPayments* value is defined as a sequence of these operational payments, where each payment is structured as:

```
{billAt:<list of intervals>,due:<value>}
```

Form 13: opPayments value

where the *billAt* value consists of a list of financial intervals that representing the the commencement of each billing cycle, while *due* indicates the number of financial intervals before these bills are due. The optional *typeSpecific* value, as in Form 11, conveys the parameters essential for *atomic* and *instrument* analytic models *type* to compute its metrics and constraints.

3.4 Analytic Model (AM)

In this section, we explain how the analytic model derives an output instance based on the input instance.

Short form	Description	Long form
$intRatio$	The number of operational intervals in each financial interval.	$in(config)(intRatio)$
ID	The set of all services IDs	$keys(in(services))$
$W(p)$	The set of all window ids in period p.	$keys((in(config)(horizon)(p)(windows)))$
$inf_{id}(s)$	The set of all inFlows id's for service s.	$keys((in(services)(s)(inFlow)))$
$outf_{id}(s)$	The set of all outFlows id's for service s.	$keys((in(services)(s)(outFlow)))$
$length(p,w)$	The length of window w at period p.	$length(in(config)(horizon)(p)(windows)(w))$
$qtyIn(id,f,p,w,i)$	The quantity of $inFlow$ f for the service with ID id during period p, window w and interval i in the input structure in.	$in(services)(id)(inFlow)(f)(p)(w)(qty)(i)$
$qtyOut(id,f,p,w,i)$	The quantity of $outFlow$ f for the service with ID id during period p, window w, and interval i in the input structure in.	$in(services)(id)(outFlow)(f)(p)(w)(qty)(i)$
$LB(id,f,p,w,i)$	The lower bound of $inFlow$ f of the service id at period p, window w, and interval i in the input structure in.	$in(services)(id)(inFlow)(f)(p)(w)(LB)(i)$
sub(cs)	The set of all subservices of the composite service cs.	in(services)(cs)(subService)
$seq(p)$	The window sequence at period p.	$config(horizon)(p)(winSeq)$
$seq(p,x)$	x^{th} window based on the window sequence during period p.	in(config)(horizon)(p)winSeq[x]
unitON(id,p,w,k)	The number of units of the service id during period p, window w, and interval i in the input structure in.	in(services)(as)(numUnitON)(p)(w)(k)
cap(as)	The capacity per unit of the service id.	in(services)(as)(capacityPerUnit)
available(id,p)	The number of available unit of service id at period p.	in(services)(id)(available)(p)
inAvailable(id)	The number of initial available unit of service id.	in(services)(id)(initAvailable)
numUnitInvested(id,p)	The number of units invested of service with ID id at period p	in(services)(id)(numUnitInvested)(p)
initLifeExp(id)	The inital life expectancy of service id.	in(services)(id)(initLifeExp)
lifeExp(id)	the life expectancy of a newly introduced service identified by id.	in(services)(id)(lifeExp)
expireIn(c, f)	The function returns the period p in which a service is set to expire, given the current period c and the specified financial interval f.	
count(p,w)	The function returns the count of windows of type w for period p present in the sequence $config(horizon)(p)(winSeq)$.	
$invPayment(id,p,i)$	The i^{th} payment if invested in service id at period p.	$in(services)(id)(payments)(invPayments)(p)(i)$
$invPayment(id,p)$	The payments associated with investing in service id during period p.	$in(services)(id)(payments)(invPayments)(p)$
$metric(m_i,id,p)$	The amount of metric m_i of service id at period p.	$out(services)(id)(metrics)(m_i)(perPeriod)(p)$
$opPayment(id,i)$	The i^{th} operational payment op of service id during period p	$in(services)(as)(payments)(opPayments)(i)$
$opPayment(id)$	The operational payments of service id during period p.	$in(services)(as)(payments)(opPayments)$
$billAt(id,op)$	The list of intervals marking the start of the billing cycle for operational payment op associated with service id.	$in(services)(id)(payments)(opPayments)(op)(billAt)$
$disDemand(ec,bp)$	The distributed demand for the energy contract (ec) at billing period (bp).	$in(services)(ec)(serviceSpecific)(disDemand)(bp)$
$demand(ec,bp)$	The demand for the energy contract (ec) at billing period (bp).	$in(services)(ec)(serviceSpecific)(demand)(bp)$
$state(v,as,p,w,k)$	The state of object v in service as at window, during period p w and interval k.	$out(as)(state)(v)(p)(w)(k)$
$degradation(ps,kwh,c,R)$	The function calculates the battery degradation based on the kWh of consumption, the current battery capacity c, and the depreciation ratio R for each cycle.	
$hourDuration(ps)$	The duration, in hours, of a battery's power supply (ps).	in(services)(ps)(serviceSpecific)(hourDuration)
$efficiency(ps)$	The efficiency of a battery's power supply (ps).	in(services)(ps)(serviceSpecific)(efficiency)

Table 1: Notations

Although the *out* and *in* structures are similar, our primary focus is on the additional components introduced in the *out* structure for both *services* and *instruments*. It's worth noting that *adjustments* in Form 1 represent the combined metrics across both *services* and *instruments*.

To calculate the other components within *out(services)*, we first consider ID to be the set comprising all service IDs. Thus, $out(services)$ is realized through a two-step process:

$$\bigcup_{id \in ID} mOut\left(periodsOut\left(in(services)(id)\right)\right)$$

The recursive function *periodsOut* processes a service input form and returns the same service, augmented with calculations for *inFlow*, *outFlow*, *constraints*, and an updated state for each interval. Meanwhile, *mOut*, another recursive function, takes the service format produced by *periodsOut* and computes metrics for each period to formulate the $out(services)$ structure.

Further information on the operations of *periodsOut* and *mOut* can be found in Sections 3.4.1 and 3.4.2, respectively. The notation employed is presented in Table 1.

3.4.1 periodOut

In this section, we demonstrate how the *periodsOut* function computes the *inFlow* and *outFlow* quantities for each interval within the windows over the investment periods. This computation includes the required constraints for both types of services: *composite (cs)* and *atomic (as)*.

***Composite* service(cs):**

Let CS be the set of all composite service IDs. For each *composite* service $cs \in CS$,we consider i as a flow in $inf_{id}(cs)$ and j as a flow in $outf_{id}(cs)$. For a given a period $p \in \{1,\ldots,P\}$, and for every window $w \in W(p)$ and interval $k \in \{1,\ldots,length(p,w)\}$, the $inFlow$ quantity qty — as defined in Form 7 — is recursively expressed as:

$$qtyIn(cs,i,p,w,k) = \sum_{s \in sub(cs)} \left(qtyIn(s,i,p,w,k) - qtyOut(s,i,p,w,k)\right)$$

Therefore, the total for each *inFlow* is given by:

$$\sum_{k=1}^{length(p,w)} qtyIn(cs,i,p,w,k)$$

As with the *inFlow* described above, the *outFlow* is expressed similarly. For each composite service $cs \in CS$, the *constraints* are formulated as a conjunction of the following constraints:

1. **Demand Constraint:** For each sub-service $s \in \text{sub}(cs)$, a flow key $i \in \{[outf_{id}(s) \cup inf_{id}(s)] - [inf_{id}(cs) \cup outf_{id}(cs)]\}$, a period $p \in \{1,\ldots,P\}$, a window $w \in W(p)$, and an interval $k \in \{1,\ldots,\text{length}(p,w)\}$, the constraint is:
$$qtyIn(s,i,p,w,k) \geq qtyOut(s,i,p,w,k).$$
2. **Bound Constraint:** To verify that the minimum/-maximum flow is achieved, the *boundConstraint* is defined as: For any $cs \in CS$, a flow key $i \in inf_{id}(cs)$, a period $p \in \{1,\ldots,P\}$, a window $w \in W(p)$, and an interval $k \in \{1,\ldots,\text{length}(p,w)\}$, the constraint is:
$$LB(cs,i,p,w,k) \leq qtyIn(cs,i,p,w,k).$$
Analogous to the lower bound (LB) constraint above, the upper bound (UB) constraints for the *outFlow* can be described similarly.
3. **Sub-service Constraint:** To ensure all constraints of sub-services are met, the *subServiceConstraints* is expressed as a conjunction of its subservices' constraints: $\bigwedge_{s \in \text{sub}(cs)} \text{out}(services)(s)(\text{constraints})$.

***Atomic* service (as):** Let AS be the set of all atomic service IDs. For each atomic service $as \in AS$, the *state* is defined as a function returning the updated state based on the previous *state* for the given service. The quantity (qty) of every *inFlow* and *outFlow* for an atomic service is computed by invoking the analytical model associated with its *type* (refer to Section 4). Furthermore, for each atomic service $as \in AS$, the *constraints* are formulated as a conjunction of the following:

1. **Bound Constraint:** Ensure adherence to both lower and upper bounds for every flow, referencing expressions similar to those detailed for composite services (see Section 3.4.1).

2. **Capacity Constraint:** Guarantee that the flow quantities do not exceed the capacity of the (ON) units for every $as \in AS$, i from $outf_{id}(as)$, $p \in \{1, \ldots, P\}$, $w \in W(p)$, and $k \in \{1, \ldots, length(p, w)\}$:

$$qtyOut(as, i, p, w, k) \leq (unitON(as, p, w, k) \times cap(as)) .$$

3. **State Consistency Constraint:** Maintain an assumption that guarantees consistent state levels at both the beginning and end of each window. Such uniformity ensures that changing the window order, either within the same period or across different periods, doesn't result in service state inconsistencies.

4. **Domain Specifics Constraints:** Incorporate any domain-specific constraints (see Section 4).

5. **Unit Count Constraints:** Ensure that the number of (ON) units for each interval does not surpass the combined count of initial and invested units retaining a viable life expectancy. To formalize this constraint, we first define the available number of units in any given period p, denoted as available(as, p). This is expressed as the summation of two quantities:

   (a) initialUnit(as, p): This represents the number of initial units that remain usable at period p:

   $$\text{initialUnit} = \begin{cases} \text{initAvailable}(as), & \text{if } p \leq \text{expireIn}(1, \text{initLifeExp}(as)) \\ 0, & \text{otherwise} \end{cases}$$

   (b) investedUnit(as, p): Denotes the sum of units, if invested in, that are still usable at period p.

   $$\text{investedUnit} = \begin{cases} \text{numUnitInvested}(as, x), & \text{if } x \leq \text{expireIn}(x, \text{lifeExp}(as)) \forall x \in \{1, \ldots, p\} \\ 0, & \text{otherwise} \end{cases}$$

   Thus,

   $$\text{available}(as, p) = \text{initialUnit}(as, p) + \text{investedUnit}(as, p)$$

   Now, unit count constraint can be expressed for every $as \in AS$, $p \in \{1, \ldots, P\}$, $w \in W(p)$, and $k \in \{1, \ldots, length(p, w)\}$:

$$unitON(as, p, w, k) \leq \text{available}(as, p)$$

3.4.2 metricOut (mOut)

In this section, we explore the functionality of the *mOut*, or *metricOut*, function. Its role is to derive metrics for both composite and atomic services by interpreting the outputs of the periodOut function across all periods. The completion of this process results in the formation of the out(service), as outlined in Form 1.

***Composite* service:** The metrics for a composite service are aggregated from the values of its subservices.

For each composite service $cs \in CS$ and throughout every period $p \in \{1, \ldots, P\}$, any *metric* m_i (e.g., cost and CO_2) is recursively expressed as:

$$metric(m_i, cs, p) = \sum_{s \in sub(cs)} metric(m_i, s, p))$$

Metrics for individual intervals within every window of a specific period are calculated similarly. The aggregate metric value for a given composite service (cs) is formulated as:

$$metricTotal(cs, m_i) = \sum_{p \in \{1, \ldots, P\}} metric(m_i, cs, p)$$

It's essential to highlight that the *cashFlow* metric is represented as a list, as indicated in Form 6. To aggregate this list, amounts occurring in the same interval are combined, producing a list with a similar form.

Furthermore, several financial metrics, which depend on the *cashFlow* (like NPV), utilize the result of *cashFlow* in conjunction with financial parameters found in $in(config)$ (e.g., intRate) to compute their values.

The *atomic* service (as): For atomic services, the metric *perPeriod* is calculated by multiplying the metric value of a given window $w \in W(p)$ with the number of times that window is repeated in the given period (p):

$$\sum_{w \in W(p)} \left(count(p,w) * \sum_{k=1}^{length(p,w)} metric(m_i,as,p,w,k) \right)$$

where *count(p,w)* is a function that counts the number of windows of type w at period p in $winSeq(p)$ sequence.

The *total* metric value is calculated by aggregating the value of a given metric over all periods as follow:

$$\sum_{p \in \{1 \ldots P\}} metrics(m_i,as,p)$$

To accurately compute the *cashFlow*, we need to consolidate both the investment and operational expenses.

To determine the specific financial *interval*—namely, the designated day set for each payment, we sum all preceding financial intervals leading up to the period p. This aggregate is then supplemented by *due*, which distinctly specifies the financial interval directly associated with period p, as follow:

$$\text{inPayment}(as, p, i)(due) + \left(\sum_{\substack{x \in \{1, \ldots, p-1\}, \\ w \in W(x)}} length(x, w) \times count(x, w) \right) \div intRatio$$

Subsequently, the *amount* corresponding to the above interval is scaled by the number of units invested during period p, described by the equation:

$$\text{inPayment}(as, p, i) \times numUnitInvested(as, p)$$

On the other hand, operational expenses typically adhere to billing periods, such as the commencement of every month, to determine the relevant charges for that duration, like maintenance costs. It's essential to understand that these expenses can be variable or static, contingent on the operation type and the service in question. For every period p within the set $\{1, \ldots, P\}$, and for every payment op from the set $opPayment(as)$, and every billing cycle b from the set $billAt(as, op)$, the interval appended to each cashFlow can be articulated as:

$$billAt(as, op)(b) + opPayment(as)(due)$$

The associated amount for the aforementioned interval is deduced using the analytical model tailored to its distinct service type, as delineated in the library (refer to Section 4).

In a parallel approach to computing $out(services)$, consider $iID = keys(in(instruments))$ as the set encompassing all instrument IDs. The output's *instrument* component, represented as *instruments*, can be delineated by:

$$\bigcup_{i \in iID} instrumentOut(in(instruments)(i))$$

In this context, the *instrumentOut* function processes an instrument input form and, upon completion, returns the instrument with its corresponding *metrics* and *constraints*. The computation methodology for *perPeriod* follows a pattern akin to the operational expenses seen with atomic services. Additionally, the metric values and constraints are derived using the analytical model corresponding to the instrument type found in the library.

4 Atomic Service and Instrument Library Models

Within the library, every atomic service is equipped with two Analytical Models (AMs). The primary AM is specifically designed to compute visibility constraints at the operational level, ensuring a balance between resource inflow and outflow over each operational interval. Once the flows are balanced over the time horizon, the secondary AM takes on the responsibility of calculating critical metrics, including cost and emissions, as it possesses visibility of the overall operational constraints. In the following, we will discuss how these AMs for each atomic service generates these metrics and constraints.

4.1 Energy Contract (ec):

The *outFlow* from the energy contract specifies the overall energy going out, so there is no inflow balancing required here. We can pull as much power as needed from the contract at the operational level within the bound constraints specified in Section 3.4.1.

The other constraints that restrict the operational outflow are calculated at the level of metrics calculation. When we calculate the bills for each billing period $b \in [v_1, ..., V_{pb}]$ specified in $billAt(ec, op)$ (e.g., every month), the AM calculates the cost by summing up the Distribution Service Charges and the Electricity Supply (ES) Service Charges. The following presents the *key:value* pairs in the *serviceSpecific* data, which are needed to calculate these charges:

```
changeInRate:[v1,...,vbp],
disDemand:[v1,...,vbp],
distDemandPWPoints:<value>,
disDemandCharge:[v1,...,vbp],
demandCharge: <value>,
demand:[v1,...,vbp],
AdjustmentCharges:AdjustmentCharges,
adjChargePWPoints:<value>,
ElectricitySupply: [v1,...,vbp],
basicCharge:<value>,
rkVAFactor:<value>,
rkVARate:<value>,
riders: <value>,
emissionFactor:[v1,...,vbp],
lineLoss: <value>
```

Form 14: (ec)serviceSpecific

These charges are the sum of the following categories:

1. Fixed Charges: These charges include fixed fees per billing period b, such as *basicCharges*.

2. Usage-Based Charges: These charges are determined by the amount of energy consumed during each billing period b. Items in this category, such as 'rides,' are calculated based on the following formula: Let $I_b = \{i_1, i_2, \ldots\}$ represent the intervals in each billing period b, W_b represent all windows within a given billing period b at investment period p, and $F = \inf_{id}(ec)$ is the set of all energy contract outflows (ec). Then, the charge for 'rides' is calculated as follows:

   $$rate \times pc(b)$$

   Here, *rate* represents the cost per kilowatt-hour (KWh) of power consumed during billing period b, and $pc(b)$ is defined as:

   $$pc(b) = \sum_{i \in I_b, f \in F, w \in W_b} qtyOut(ec, f, p, w, i)$$

3. Demand Charges: These charges are based on the highest amount of power used during an operational interval (e.g., 30 minutes). To account for this, we add the following constraint for each $demand(ec, b)$ in $\{v_1, \ldots, v_{bp}\}$:

   $$demand(ec, b) \geq pc(b)$$

   where b is a billing period, and bp is the total number of billing periods in the time horizon.

4. Peak Charges: This charge structure results in higher prices during peak hours. In this context,

'demand' is constrained to exceed the average kilowatts (kW) measured during specific peak intervals, which can vary based on time, days of the week, or seasons.

5. Rate Tiers: The charges depend on rate structures with different tiers. The criteria for moving between tiers depend on factors such as "disDemand" and "demand." These transitions between tiers are defined by specific capacity thresholds, which are specified as "piecewise points" in *"disDemandPWPoints"* and *"adjChargePWPoints"* respectively.

6. Rolling Average Charges: This refers to a billing method that takes into account the average energy consumption over a certain period, which can change over time as new data becomes available. It's typically used to smooth out billing fluctuations and may involve looking at energy usage over a rolling average of several months, such as the previous 11 months. Given this, for every outFlow f outf$_{id}(ec)$ and for every interval k in the set $\{1, \ldots, \text{length}(p, w)\}$ within each window $w \in W(p)$ in a given rolling month of the specified period p, the outFlow quantity is constrained by: disDemand$(ec, bp) \geq$ qtyOut(ec, f, p, w, k).

For each billing period, the AM calculates the actual emissions (Scope 2) by multiplying the emission factor with the actual power consumption (the *outflow* in a given billing period), factoring in the line loss. It's important to note that the aforementioned costs are updated in the *cashFlow*, as per the structure outlined in Form 6. Additionally, the charge rates (e.g., basicCharge,...) conform to the rate increments delineated for every billing period in *changeInRate*.

4.2 Power Storage (ps):

The following presents the *key:value* pair in the *serviceSpecific* which is needed to calculate the operation, maintenance and investment cost, emissions and as well as adjust the capacity of the battery to account for battery degradation:

```
mCost:[v_1,...,v_bp],
batteryCapacity:[v_1,...,v_p],
capacityPerUnit:<value>,
MaxNumOfUnit: <value>,
efficiency:<value>,
co2Factor: <value>,
hourDuration:<value>,
```

Form 15: (ps)serviceSpecific

The power storage AM use the quantities of power *inFlow* and *outFlow* at every operational interval k and the *state* of the battery charge (chargeLevel) to calculate new state as follow:

$$\begin{aligned} &state(chargeLevel, ps, p, w, k) \\ &\quad = state(chargeLevel, ps, p, w, k-1) \\ &\qquad + qtyIn(ps, inf_{id}(ps), p, w, k) \\ &\qquad - (qtyOut(ps, outf_{id}(ps), p, w, k) * efficiency(ps)) \end{aligned}$$

The AM also incorporates a constraint to maintain the balance of charge levels at each interval through either charging or discharging operations. The constraints are as follows:

$$qtyIn(ps, inf_{id}(ps), p, w, k) >= 0$$

$$qtyOut(ps, inf_{id}(ps), p, w, k) >= 0$$

$$qtyIn(ps, inf_{id}(ps), p, w, k) <= Charge * M$$

$$qtyOut(ps, inf_{id}(ps), p, w, k) <= (Charge - 1) * M$$

Here, Charge is a binary variable, and M is a large positive number. It's important to note that the combined charging level across all available units in period p must be bounded between zero and the total battery capacity. This can be mathematically represented as

$$0 \leq state(chargeLevel, ps, p, w, k) \leq totalCapcity(ps, p) \tag{3}$$

The total capacity is updated each period to account for newly invested units as well as the degradation of batteries purchased in previous periods. It's assumed that degradation impacts all units uniformly. This can be mathematically represented as:

$$\begin{aligned} &totalCapcity(ps, p+1) \\ &\quad = capacityPerUnit(ps) * hourDuration(ps) \\ &\qquad * investedNumUnit(ps, p+1) \\ &\qquad + degradation(ps, Kwh, totalCapcity(ps, p), R) \end{aligned}$$

4.3 Gas Boiler (gb):

The system calculates the heat output by taking into account the efficiency of the gas boiler when provided with a specific quantity of gas. The cost associated with operating the gas boiler during each billing period is tied to gas consumption, resembling the Usage-Based Charges in the energy contract (*ec*). Additionally, this cost is affected by both fixed and variable maintenance expenses. The emissions generated by this process are determined by a specific emissions factor associated with burning gas.

4.4 Electric Boiler (eb):

Similar to the gas boiler, in this case, we use electricity as the input, and we have already taken into account its cost and emissions. Therefore, there is no need to introduce additional costs or emissions to avoid double-counting. The only cost considered is for maintenance.

4.5 Solar Panel(sp):

The model calculates the kWh output by taking into account the solar radiation at each interval and the specific type of solar panel in use. The cost associated with maintaining these panels is determined by the quantity of units available during any given period.

4.6 Transformer (t)

The transformer is classified as an instrument because it lacks operational control. We assume that the transformer upgraded at most once throughout the time horizon. This is done if we invest in a service that require electricity, such as electric boilers. The mathematical representation is:

$$available(t, p) * UB \geq available(eb, p)$$

$$0 \leq numUnitInvested(t, p) \leq 1$$

Here, UB denotes the upper limit of units the transformer can handle. The associated cost is updated in the *cashFlow*, akin to other services.

4.7 RECs:

The following presents the *key:value* pair in the *serviceSpecific* which is needed to calculate the cost and emissions reduction for RECs purchased:

```
costMwh:[v1,...,vm],
co2Factor:<value>,
adjCO2:[v1,...,vap],
RetiredRECs: [v1,...,vap],
investedNumRECs:[v1,...,vbp]
```

Form 16: (RECs)serviceSpecific

To ensure accurate accounting for RECs within a defined accounting period, typically a year, there are specific timelines set for recognizing these certificates. Specifically, these RECs can either be generated within the accounting period, in the six months preceding it, or the three months following it. This flexible window for REC recognition is implemented to offer more latitude in REC purchasing. To model this, we introduce the following parameter:

- The cost per Mwh *costMwh* for each month m.
- The adjusted CO2 *adjCO2*: This refers to the amount of CO2 emissions offset due to the utilization of these RECs within the accounting period (year).
- *RetiredRECs*: This represent the RECs that have been claimed within each accounting period (ap).
- *investedNumRECs*: This represents the actual month when the RECs were procured, denoted as bp.

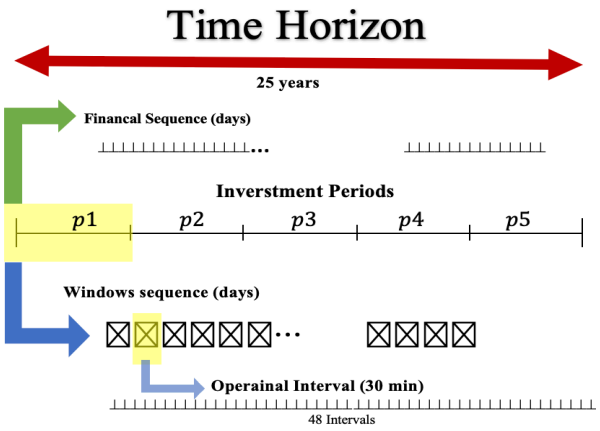

Figure 5: Time horizon

This facilitates a billing mechanism based on the monthly cost delineated in *costMwh*, multiplied by the corresponding *InvestedNumRECs* of that month. However, when it comes to actual emission offset, it is aligned with the month specified by *RetiredRECs*. Importantly, this system accommodates a flexible window. Given this setup, it's essential to maintain equilibrium between the aggregates of *investedNumRECs* and *RetiredRECs* throughout the entire period. Consequently, these costs are integrated into the *cashFlow*, and the CO2 metrics are adjusted accordingly.

4.8 Offset:

The AM follows a structure similar to that of RECs, but the primary emphasis here is on the cost per ton of CO2. This model can be applied to address both scope 1 and scope 2 emissions, while RECs predominantly cater to scope 2 emissions related to contractual energy sources.

5 Experimental study

In this section, we conducted two experimental studies to assess the system's scalability and its precision in addressing real-world investment problems. Our experiments were conducted on a batch-processing cluster, leveraging a single core of the AMD Opteron Processor 6276. For optimization purposes, we employed CPLEX 22 as our primary tool.

In the first experimentation we assess the system's ability to optimize investments for large-scale problems using exptended time horizon with differnt level of carbin constraints. In our study, we devised four scenarios using a grid system with modules like solar panels, energy contracts, batteries, electric and gas boilers, transformers, RECs, and offsets. Our goal was to optimize operational control strategies and infrastructure expansion over a

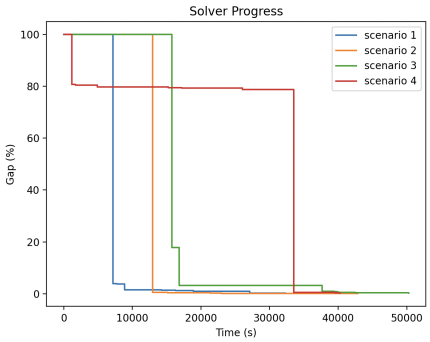

Figure 6: CPLEX solution progress

25-year investment horizon, divided into five-year periods with 30-minute operational intervals. We used preprocessing algorithms with an epsilon of 300 kW to generate windows. The second window splitting had a bandwidth of 0.25, with each window representing a day. We incorporated business constraints to meet specific carbon reduction targets. Our primary objective was to minimize the system's present value cost (PVC) throughout the investment horizon.

In our experimental scenarios, the objective was to determine the most efficient financial investment strategy by minimize the system's present value cost (PVC) under varying carbon neutrality constraints. The stopping criteria were set either at a time limit of 5 days or upon reaching a gap of 0.3%.

Here are our findings:

1. **No Carbon Reduction Constraint**: In this scenario, where no restrictions were placed on carbon emissions, the first feasible solution appeared at 75.73 seconds with a 100% gap (see Figure 6). As the computations progressed, the solution narrowed to a gap of 0.28% after 7.5 hours, reflecting an absolute gap of $358,722 out of $127,813,907.

2. **Carbon Neutrality by the Fourth Period**: When we introduced a carbon neutrality constraint effective from the fourth period onward, the model took longer to find its first feasible solution – 950.24 seconds with a starting gap of 100%. With extended computation, the gap reduced to 0.09% after 8 hours, corresponding to an absolute gap of $134,694 out of %136,212,897.

3. **Achieving Carbon Neutrality by the Second Period**: By advancing the carbon neutrality constraint to start from the second period, the solution achieved a gap of 0.29% after 14.5 minutes, with an absolute gap of $478,873 out of $ 160,173,051.

4. **Mandatory Carbon Neutrality Across All Periods**: In our strictest scenario, which mandated carbon neutrality across all periods, the model identified a solution within 77.96 seconds, albeit with a gap of 80.14%. With extended processing, this gap diminished significantly, stabilizing at 0.26% after 11 hours. This translates to an absolute gap of $ 443,671 from a total of $ 168,774,499.

To summarize, these trials served as preliminary investigations into the viability of using our model to tackle realistic investment scenarios. The results are promising, as in each case, the model was able to converge to a solution near the optimum within a feasible timeframe.

In our second experimentation, the focus was on generating representative windows and evaluating how accurately these representations compare to the actual windows. In this experiment, we formulated a problem that incorporated three modalities representing the electricity contract, boiler, and solar energy – the latter emulating the supply from solar panels. We utilized data from George Mason University's Fairfax campus, specifically the electricity demand in 30-minute intervals throughout the year 2019. Additionally, heat demand data for the same campus, with similar granularity, was used, along with solar radiation estimates sourced from the National Renewable Energy Laboratory for the corresponding region. First, we optimized both the operational and investment controls for the actual operation windows. Subsequently, we repeated the experiment using the same data, but reduced the number of windows by leveraging our preprocessing algorithm. This generated 54 representative windows with an epsilon of 30 and bandwidth of 0.25, as well as 16 representative windows using an epsilon of 300 and bandwidth of 30, respectively. Table 2 shows the differences in key performance indicators between the actual data and the representative windows, as well as the percentage of error in each run compared to the actual windows. From the table, it's evident that the present value cost for both representative problems was minimal. This suggests that, at least for the modalities we employed, the generated representative windows are a good indicator for the actual windows, and the preprocessing algorithm provides a solid approximation of the actual data. This assessment serves as an initial evaluation of the algorithm. Further experimentation could be pursued to generalize our findings.

Number of Windows	365 (Actual)	57 (Representative)	Error	16 (Representative)	Error
Present Value Cost	10,534,371	10,535,022	0.01%	10,568,506	0.32%
Operation Cost	6,822,388	6,822,651	0.00%	6,822,651	0.00%
CO_2e (scope1)	13,609	13,609	0.00%	13,609	0.00%
CO_2e (scope2)	25,778	25,778	0.00%	25,778	0.00%
Electric Contract (PVC)	5,873,315	5,874,133	0.01%	5,908,495	0.60%
Boilers (PVC)	4,655,862	4,655,695	0.00%	4,654,818	0.02%
Solar (PVC)	5,192	5,192	0.00%	5,192	0.00%

Table 2: Actual windows vs representative windows

6 Conclusion and Future Work

As part of GADGET development, we enhanced the integration of financial instruments within the SNIM model, modeled various micro-grid components with a focus on operational and financial aspects, established a pre-processor engine to streamline the process, and conducted two empirical studies. The first study showcased GADGET's effectiveness in addressing large-scale investment challenges, while the second validated the accuracy of its pre-processing method in faithfully representing real-world supply and demand patterns.

Future work will concentrate (1) expanding the extensible library of domain-specific analytic models relevant to system components, including models for building retrofits, energy efficiency retrofits, central utility electrification, and distributed energy sources; (2) conducting a case study to assess GMU's journey toward achieving carbon neutrality by 2040; and (3) enhancing the time complexity associated with the optimization problem in the model by integrating a pre-processing algorithm. This algorithm is designed to break down the challenges posed by the analytical model into smaller, pre-resolved operational problems.

7 Acknowledgment

These experiments were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA. (URL: http://orc.gmu.edu)

References

[1] W. Wei, W. Danman, W. Qiuwei, M. Shafie-Khah, and J. P. Catalao, "Interdependence between transportation system and power distribution system: A comprehensive review on models and applications," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 3, pp. 433–448, 2019.

[2] X. Xin, X. Wang, L. Ma, K. Chen, and M. Ye, "Shipping network design–infrastructure investment joint optimization model: a case study of west africa," *Maritime Policy & Management*, vol. 49, no. 5, pp. 620–646, 2022.

[3] Y. Ru, J. Kleissl, and S. Martinez, "Storage size determination for grid-connected photovoltaic systems," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 68–81, 2013.

[4] Z. Wang, B. Chen, J. Wang, J. Kim, and M. M. Begovic, "Robust optimization based optimal dg placement in microgrids," *IEEE Transactions on Smart Grid*, vol. 5, no. 5, pp. 2173–2182, 2014.

[5] J. M. Home-Ortiz, O. D. Melgar-Dominguez, M. Pourakbari-Kasmaei, and J. R. S. Mantovani, "A stochastic mixed-integer convex programming model for long-term distribution system expansion planning considering greenhouse gas emission mitigation," *International Journal of Electrical Power and Energy Systems*, vol. 108, pp. 86,95, 2019.

[6] K. Liaqat, "Modeling, optimization, and software development for concentrated solar power plants," 2021.

[7] M. T. Al-Nory, "Optimal decision guidance for the electricity supply chain integration with renewable energy: Aligning smart cities research with sustainable development goals," *IEEE Access*, vol. 7, pp. 74 996–75 006, 2019.

[8] M. Dicorato, G. Forte, M. Pisani, and M. Trovato, "Planning and operating combined wind-storage system in electricity market," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 2, pp. 209–217, 2012.

[9] B. Alyahya and A. Brodsky, "A decision guidance system for optimal operation of hybrid power desalination service network." in *ICORES*, 2021, pp. 416–424.

[10] M. Breen, J. Upton, and M. Murphy, "Development of a discrete infrastructure optimization model for economic assessment on dairy farms (diomond)," *Computers and Electronics in Agriculture*, vol. 156, pp. 508 – 522, 2019.

[11] Q. Fu, L. F. Montoya, A. Solanki, A. Nasiri, V. Bhavaraju, T. Abdallah, and D. C. Yu, "Microgrid generation capacity design with renewables and energy storage addressing power quality and surety," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2019–2027, 2012.

[12] T. Kolster, R. Krebs, S. Niessen, and M. Duckheim, "The contribution of distributed flexibility potentials to corrective transmission system operation for strongly renewable energy systems," *Applied energy*, vol. 279, pp. 115 870–, 2020.

[13] "Reports from federal university advance knowledge in industrial engineering (a robust optimization approach for cash flow management in stationery companies)," pp. 1074–, 2016.

[14] T.-Y. Hsieh and H.-L. Liu, "Genetic algorithm for optimization of infrastructure investment under time-resource constraints," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 3, pp. 203–212, 2004.

[15] T. Lambert, P. Gilman, and P. Lilienthal, "Micropower system modeling with homer," *Integration of alternative sources of energy*, vol. 1, no. 1, pp. 379–385, 2006.

[16] B. Alyahya and A. Brodsky, "A decision guidance system for optimal infrastructure investments," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2021, pp. 1337–1342.

[17] M. O. Nachawati, A. Brodsky, and J. Luo, "Unity decision guidance management system: Analytics engine and reusable model repository." in *ICEIS (1)*, 2017, pp. 312–323.

[18] A. Brodsky and X. S. Wang, "Decision-guidance management systems (dgms): Seamless integration of data acquisition, learning, prediction and optimization," in *Proceedings of the 41st annual Hawaii international conference on system sciences (HICSS 2008)*. IEEE, 2008, pp. 71–71.

[19] A. Brodsky and J. Luo, "Decision guidance analytics language (dgal)," in *Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 1*, ser. ICEIS 2015. Portugal: SCITEPRESS - Science and Technology Publications, Lda, 2015, pp. 67–78.