HouseCompanyEnergyModel.jl

EasyDR project

Helmi Hankimaa 01.09.2022

Contents

1 Introduction

The purpose of the HouseCompanyEnergyModel.jl package is to provide the ground work for modeling energy related decision making of a house company. The package was created as part of the EasyDR project.

The package is a distilled version of the Predicer model developed at VTT. Currently, the formulation merely optimises the production and allocation of energy for a set of scenarios and minimises the expected costs as the objective function. By allocation of energy, we mean that energy the house company produces can be either used by the house company, sold to the energy market or stored. However, notice that the current formulation assumes perfect foresight and treats the scenarios independently. Meaning that, the model does not have two stage decision making as it does not have non-ancipativity constraints. Therefore, at the moment the package only provides the ground work for modeling more interesting decision making problems, which will require changing the formulation. An interesting premise to optimise would be the use of an electric car fleet as a form of energy storage for the house company.

The contents of this documentation include a description of the structure and formulation of the model and two simple examples. The first example is a very simple example of a house company which has a solar panel. The purpose of the example is to illustrate the usage of the package. The second example was the initial use case of HouseCompanyEnergyModel.jl, which was to test the logic of a set of bidding constraints. With the given example, these constraints were rebutted.

2 System Structure

The model consists of three types of elements: nodes, processes, and flows. The flows form links between nodes and processes.

2.1 Nodes

Node elements represent different energy types in the model. Examples of nodes could be electricity, district heat or hydrogen. There are four subcategories of nodes:

1. Energy node. This is the basic node type. These nodes represent energy in the system. Energy balance is maintained over these nodes and they can be subject to external flow representing inflow or demand on the node from outside of the model's scope.

2. Storage node. These are energy nodes which have storage capability. They can store energy up to a capacity value. Just like with energy nodes, energy balance is maintained over these nodes and they can be subject to external flow representing inflow or demand on the node from outside of the model's scope.

3. Commodity node. This is a special type of node which represents commodities bought from outside of the model's scope. Commodities are associated with a cost, which is declared as a parameter of the node. Energy balance is not maintained over these nodes.

4. Market node. This is a special type of node which represent energy markets that the house company is involved in. In the model, the house company gains profits and incurs costs from energy transferred from and to a market node. The market has a predicted price estimate, which is declared as a parameter of the node. Energy balance is not maintained over these nodes.

In the implementation, there exists a `struct` for each node type. These are called `EnergyNode`, `StorageNode`, `CommodityNode` and `MarketNode`. These all share the same super type `AbstractNode`. All these types have their own constructor functions which do some parameter value validation. For instance, the constructor functions check the dimensions of the time series parameters. They also check that parameter values fall within their feasible regions outlined in Table 3. The constructor functions are `energy_node()`, `storage_node()`, `commodity_node()` and `market_node()`.

2.2 Processes

Processes represent energy generating or transforming units. Examples of processes could be a hydroelectric system or a solar panel. There are three different types of processes in the model.

1. Flexible process. This type of processes represent energy generation technologies where the operating load can be ramped between zero load and full capacity continuously. An efficiency factor determines the relationship between input and output flows. These processes also have a capacity factor time series which can be used to model maintenance schedules or other varying restrictions on available capacity.

2. Online process. This type of process has online offline functionality. When it is online, the process must run at a minimum load. Furthermore, they have a cost associated with starting the unit and minimum online and offline period lengths. Similarly, to flexible processes, efficiency is considered for input and output flows and these processes also have a capacity factor time series which can be used to model maintenance schedules or other varying restrictions on available capacity.

3. Variable renewable energy (VRE) process. This is a special process type for which the generation capacity is dependent on exogenous uncertainty. This uncertainty is modeled by a capacity factor time series which represents a prediction of available solar, wind or tidal energy. These processes do not have inflows as the energy source comes from outside the model's

scope. Therefore, these processes are also not associated with an efficiency factor.

In the implementation, there exists a `struct` for each process type. These are called `FlexibleProcess`, `OnlineProcess` and `VREProcess`. These all share the same super type `AbstractProcess`. All these types have their own constructor functions which do some parameter value validation. For instance, the constructor functions check the dimensions of the time series parameters. They also check that parameter values fall within their feasible regions outlined in Table 3. The constructor functions are `flexible_process()`, `online_process()` and `vre_process()`.

2.3 Flows

Flows represent transfer of energy between nodes and processes. There are three types of flows:

1. Process flows move energy between nodes and processes. Different aspects of processes are modeled by creating constraints on their process flows. The process flows are constrained by efficiency, capacity limits and ramp rate limits. These flows can have VOM costs.

2. Transfer flows move energy without losses between two nodes. These flows are unconstrained and their use does not incur costs.

3. Market flows transfer energy without losses from node A to node B (or from node B to node A) where either node A or node B is a market node and the other node is an energy or storage node. For a market node, both directions of flow (into the node and out of the node) should be established. These flows are unconstrained and their use does not incur costs.

In the implementation, there exists a `struct` for each flow type. These are called `ProcessFlow`, `TransferFlow` and `MarketFlow`. These all share the same super type `AbstractFlow`. All these types have their own constructor functions which do some parameter value validation. For instance, the constructor functions check the dimensions of the time series parameters. They also check that parameter values fall within their feasible regions outlined in Table 3. The constructor functions are `process_flow()`, `transfer_flow()` and `market_flow()`. Notice that the function `market_flow()` returns two `MarketFlow` structures: one for each direction of transfer between the market node and other node.

3 Model notation

3.1 Sets

Table 1: Names and descriptions of sets in the model.

Name	Description
$t \in T$	Time steps
$s \in S$	Scenarios
N	Nodes
N^{energy}	Subset of nodes N that are energy nodes.
$N^{storage}$	Subset of nodes N that have storage.
$N^{commodity}$	Subset of nodes N which represent a commodity.
N^{market}	Subset of nodes N which represent markets.
P	Processes
$P^{flexible}$	Subset of processes P that are flexible processes.
P^{vre}	Subset of processes P that are VRE generation.
P^{online}	Subset of processes P that have online functionality.
$(i,j) \in F$	Set of flows
$(i,j) \in F^{process}$	Subset of flows F which are connections between a node and process.
$(i,j) \in F^{transfer}$	Subset of flows F which are connections between two nodes.
$(i,j) \in F^{market}$	Subset of flows F which are connections between an energy or storage node and market node.

3.2 Decision variables

The decision variables are outlined in Table 2. In the implementation, the variables are declared using functions `flow_variables()`, `state_variables()`, `shortage_surplus_variables()` and `start_stop_online_variables()`. These functions add the variables to the model declare their feasible regions as stated in Table 2.

Table 2: Names, descriptions and feasible regions of decision variables.

Variable	Description	Feasible region
$v_{i,j,s,t}^{flow}$	Flow from $i \in \{N, P\}$ to $j \in \{N, P\}$ in scenario $s \in S$ and time $t \in T$. Units in MW.	≥ 0
$v_{n,s,t}^{flowShortage}$	Slack variable for node $n \in \{N^{energy}, N^{storage}\}$ if there is a shortage of energy flow into the node. Units in MW.	≥ 0
$v_{n,s,t}^{flowSurplus}$	Slack variable for node $n \in \{N^{energy}, N^{storage}\}$ if there is a surplus of energy flow into the node. Units in MW.	≥ 0
$v_{p,s,t}^{online}$	Indicator variable for process $p \in P^{online}$ being online at time t in scenario s. Note that $1 =$ on and $0 =$ off.	$\{0, 1\}$
$v_{p,s,t}^{start}$	Indicator variable for process $p \in P^{online}$ being turned online at time t in scenario s.	$\{0, 1\}$
$v_{p,s,t}^{stop}$	Indicator for process $p \in P^{online}$ being turned offline at time t in scenario s.	$\{0, 1\}$
$v_{n,s,t}^{state}$	State variable for node $n \in N^{storage}$ in scenario s and time t. Units in MWh.	$[0, d^{stateMax}]$

3.3 Parameters

Table 3: Names, descriptions and range of values for state, flow, process and cost parameters.

Name	Description	Range of values
$d_n^{stateMax}$	Maximum storage capacity of node $n \in N^{storage}$. Units in MWh.	≥ 0
$d_n^{initialState}$	Initial state of node $n \in N^{storage}$.	$\leq d_n^{stateMax}$
$d_n^{stateLoss}$	Loss factor of storage of node $n \in N^{storage}$.	[0,1]
$d_n^{flowMaxIn}$	Maximum flow capacity into node $n \in N^{storage}$ (charging) . Units in MW.	≥ 0
$d_n^{flowMaxOut}$	Maximum flow capacity out of node $n \in N^{storage}$ (discharging). Units in MW.	≥ 0
$d_n^{flowExternal}$	External inflow or demand of node $n \in \{N^{energy}, N^{storage}\}$.	$\in \mathbb{R}$
$d_{i,j}^{flowCapacity}$	Maximum flow capacity through link $(i,j) \in F$.	≥ 0
$d_p^{minLoad}$	Minimum fraction of capacity to be used at all times if process $p \in P^{online}$ is online.	[0,1]
$d_p^{minOnline}$	Minimum online period of process $p \in P^{online}$.	$\in \mathbb{Z}^+$
$d_p^{minOffline}$	Minimum offline period of process $p \in P^{online}$.	$\in \mathbb{Z}^+$
$d_p^{initialStatus}$	Initial status of online process $p \in P^{online}$. Notice that 1 = on and 0 = off.	{0,1}
$d_{p,s,t}^{efficiency}$	Throughput efficiency of process $p \in \{P^{flexible}, P^{online}\}$.	[0,1]
$d_{p,s,t}^{cf}$	Upper bound flow capacity factor for process $p \in P$.	[0,1]
$d_p^{rampRate}$	Ramp rate factor for process $p \in \{P^{flexible}, P^{online}\}$.	[0,1]
$c_{n,s,t}^{comm}$	Price of commodity at node $n \in N^{commodity}$. Unit e/MWh.	≥ 0
$c_{n,s,t}^{energy}$	Energy market price at node $n \in N^{market}$. Unit e/MWh.	≥ 0
$c_{i,j}^{vom}$	VOM cost associated with process flow through $(i,j) \in F^{process}$.	≥ 0
c_p^{start}	Start cost of process $p \in P^{online}$.	≥ 0
$c^{penalty}$	Cost of slack energy.	≥ 0
π_s	Probability of scenario s.	[0,1]

4 Model formulation

4.1 Objective function

The objective is to minimise the total costs.

$$\begin{aligned} \textit{total costs} = {} & \textit{commodity cost} + \textit{market cost - market profit} \\ & + \textit{VOM cost} + \textit{start cost} + \textit{penalty cost} \end{aligned} \tag{1}$$

where the terms are defined as follows

$$\textit{commodity cost} = \sum_{\substack{(i,j)\in F: \\ i\in N^{commodity}}} \sum_{s\in S} \sum_{t\in T} \pi_s \; c^{comm}_{i,s,t} \; v^{flow}_{i,j,s,t} \tag{2}$$

$$\textit{market cost} = \sum_{\substack{(i,j)\in F: \\ i\in N^{market}}} \sum_{s\in S} \sum_{t\in T} \pi_s \; c^{energy}_{i,s,t} \; v^{flow}_{i,j,s,t} \tag{3}$$

$$\textit{market profit} = \sum_{\substack{(i,j)\in F: \\ j\in N^{market}}} \sum_{s\in S} \sum_{t\in T} \pi_s \; c^{energy}_{j,s,t} \; v^{flow}_{i,j,s,t} \tag{4}$$

$$\textit{VOM cost} = \sum_{(i,j)\in F^{process}} \sum_{s\in S} \sum_{t\in T} \pi_s \; c^{vom}_{i,j} \; v^{flow}_{i,j,s,t} \tag{5}$$

$$\textit{start cost} = \sum_{p\in P^{online}} \sum_{s\in S} \sum_{t\in T} \pi_s \; c^{start}_{p} \; v^{start}_{p,s,t} \tag{6}$$

$$\textit{penalty cost} = \sum_{n\in\{N^{energy}, N^{storage}\}} \sum_{s\in S} \sum_{t\in T} \pi_s \; c^{penalty} \; (f^{surplus}_{n,s,t} + f^{shortage}_{n,s,t}) \tag{7}$$

In the implementation, function `declare_objective()` add the objective function to the JuMP model.

4.2 Storage constraints

The state of a storage node, represents how much energy is currently in the node's storage. The following constraints apply for all storage nodes in all scenarios and time steps. Notice that when $t = 1$, we use $v^{state}_{n,s,t-1} = d^{initialState}_{n}$.

$$0 \leq v_{n,s,t}^{state} \leq d_{n}^{stateMax} \tag{8}$$

$$-d_{n}^{flowMaxOut} \leq v_{n,s,t}^{state} - (1 - d_{n}^{stateLoss})\ v_{n,s,t-1}^{state} \tag{9}$$

$$d_{n}^{flowMaxIn} \geq v_{n,s,t}^{state} - (1 - d_{n}^{stateLoss})\ v_{n,s,t-1}^{state} \tag{10}$$

$$\forall n \in N^{storage}, s \in S, t \in T$$

In the implementation, Constraints (9) and (10) are called charging and discharging constraints and declared in function `charging_discharging_constraints()`. Constraint (8) is added when declaring the state variables in function `state_variables()`.

4.3 Energy flow constraints

The constraints in this section define the lower and upper bounds for flow variables. All flow variables must take positive values. Constraint (11) enforces this.

$$v_{i,j,s,t}^{flow} \geq 0 \qquad \forall (i,j) \in F, s \in S, t \in T \tag{11}$$

The upper bound for flows connected to a flexible process is the capacity of the link. This is enforced with Constraint (12).

$$v_{i,j,s,t}^{flow} \leq d_{i,j}^{flowCapacity} \tag{12}$$

$$\forall (i,j) \in F : i \in P^{flexible} \vee j \in P^{flexible}, s \in S, t \in T$$

Constraints (13) and (14) apply to flows that are connected to an online process. Notice that variable $v_{p,s,t}^{online}$ is a binary indicator for the online status of the process and parameter $d_{p}^{minLoad} \in [0,1]\ \forall p \in P^{online}$. Furthermore, note that Constraint (13) tightens the lower bound of Constraint (11) for these flow variables.

$$v_{i,j,s,t}^{flow} \geq d_{p}^{minLoad}\ d_{i,j}^{flowCapacity}\ v_{p,s,t}^{online} \tag{13}$$

$$v_{i,j,s,t}^{flow} \leq d_{i,j}^{flowCapacity}\ v_{p,s,t}^{online} \tag{14}$$

$$\forall (i,j) \in F : i = p \in P^{online} \vee j = p \in P^{online}, s \in S, t \in T$$

In the implementation, Constraints (12), (13), (14) are called process flow constraints and are declared in function `process_flow_constraints()`. The zero lower bound seen in Constraint (11) is added when declaring the flow variables in `flow_variables()`.

The upper bounds for flows connected to VRE processes is defined in the next subsection.

Notice that transfer and market flows $(i,j) \in \{F^{transfer}, F^{market}\}$ do not have an upper bound, since the model assumes that the transmission system operator (TSO) will offer the capacity needed.

4.3.1 Capacity factor flow constraints

In the model, processes are associated with a capacity factor (CF), which creates an upper bound on their output flow. The CF time series models exogenous factors on the available capacity of the process such as maintenance schedules or availability of solar and wind energy. Notice that the parameter values $d^{cf}_{p,s,t} \in [0,1]\ \forall p \in P, s \in S, t \in T$.

Since VRE processes do not have input flows, the following constraint establishes the upper bound for all flows connected to a VRE processes. For online and flexible processes, this constraint is only applied to the output flows of the process and it may override Constraint (12) or (14) that was previously declared for these variables. Notice that the efficiency constraint (20) introduced later relates this constraints' effect to the input flows of the process as well.

$$v^{flow}_{i,j,s,t} \leq d^{cf}_{i,s,t}\ d^{flowCapacity}_{i,j} \quad \forall (i,j) \in F : i \in P, s \in S, t \in T \tag{15}$$

In the implementation, Constraint (15) is added to the model using function `cf_flow_constraints()`.

4.4 Ramp rate constraints

The ramp rate of a process defines how fast its operative load can be changed from one time step to the next. Ramp rate constraints apply to all flows connected to flexible and online processes. The constraints apply to all time steps with the exception of the first time step. Notice that the pairs of constraints are ordered so that the first one constrains ramping up the process and the second one constrains ramping down the process. Constraints (16) and (17) apply to flexible processes.

$$v^{flow}_{i,j,s,t} - v^{flow}_{i,j,s,t-1} \leq d^{rampRate}_{p}\ d^{flowCapacity}_{i,j} \tag{16}$$

$$v^{flow}_{i,j,s,t-1} - v^{flow}_{i,j,s,t} \leq d^{rampRate}_{p}\ d^{flowCapacity}_{i,j} \tag{17}$$

$$\forall (i,j) \in F : i = p \in P^{flexible} \vee j = p \in P^{flexible}, s \in S, t \in T \setminus \{1\}$$

The following constraints apply to processes with online functionality. The maximisation on the right hand side of the inequality ensures that the process can be turned off or on in one time step even if its ramp rate per time step is less than its minimum operative load.

$$v_{i,j,s,t}^{flow} - v_{i,j,s,t-1}^{flow} \\ \leq d_p^{rampRate} + \max\{0, d_p^{minLoad}\ d_{i,j}^{flowCapacity} - d_p^{rampRate}\} v_{p,s,t}^{start} \tag{18}$$

$$v_{i,j,s,t-1}^{flow} - v_{i,j,s,t}^{flow} \\ \leq d_p^{rampRate} + \max\{0, d_p^{minLoad}\ d_{i,j}^{flowCapacity} - d_p^{rampRate}\} v_{p,s,t}^{stop} \tag{19}$$

$$\forall (i,j) \in F : i = p \in P^{online} \vee j = p \in P^{online}, s \in S, t \in T \setminus \{1\}$$

In the implementation the ramping constraints (16)-(19) are added to the model using function `process_ramp_rate_constraints()`.

4.5 Process efficiency constraints

In the model, the output flow of flexible and online processes is determined by their input flows and the efficiency of the process. This is modeled using Constraint (20), where the efficiency factor $d_{p,s,t}^{efficiency}$ defines how the sum of the input flows relates to the sum of output flows of the process.

$$\sum_{\substack{(i,j)\in F: \\ i=p}} v_{i,j,s,t}^{flow} = d_{p,s,t}^{efficiency} \sum_{\substack{(i,j)\in F: \\ j=p}} v_{i,j,s,t}^{flow} \tag{20}$$
$$\forall p \in \{P^{flexible}, P^{online}\}, s \in S, t \in T :$$

In the implementation, Constraint (20) is called the efficiency constraint and it is declared using function `process_efficiency_constraints()`.

4.6 Online functionality constraints

Processes with online functionality have binary online variables $v_{p,s,t}^{online}$ which follow the dynamic Equation (21). The status of online processes are changed via binary start variables $v_{p,s,t}^{start}$ and stop variables $v_{p,s,t}^{stop}$. Furthermore, the minimum lengths of the online and offline periods are enforced Constraints (22) and (23). These constraints, state that $v_{p,s,t}^{online}$ must be 1 for the minimum online period after the process is started and it must be 0 for the minimum offline period after the process is stopped.

$$v_{p,s,t}^{online} = v_{p,s,t-1}^{online} + v_{p,s,t}^{start} - v_{p,s,t}^{stop} \tag{21}$$

$$v_{p,s,t'}^{online} \geq v_{p,s,t}^{start} \qquad t \leq t' \leq \min\{t^{max}, t + d_p^{minOnline}/\tau\} \tag{22}$$

$$v_{p,s,t'}^{online} \leq v_{p,s,t}^{stop} \qquad t \leq t' \leq \min\{t^{max}, t + d_p^{minOffline}/\tau\} \tag{23}$$

$$\forall p \in P^{online}, s \in S, t \in T:$$

In the implementation Constraints (21)-(23) are added to the model using function `online_functionality_constraints()`.

4.7 Energy balance constraints

Energy balance over energy and storage nodes is enforced using Constraints (24) and (25). The balance equations consist of terms related to:

- change in state between adjacent time periods (only applicable to storage nodes)
- difference in input and output flow of the node,
- external flow which models inflow (positive) or demand (negative),
- slack variables called shortage and surplus of flow.

The following constraint applies to energy nodes.

$$0 = \sum_{(i,j) \in F:\ j=n} v_{i,j,s,t}^{flow} - \sum_{(i,j) \in F:\ i=n} v_{i,j,s,t}^{flow}$$

$$+ d_n^{flowExternal} + v_{n,s,t}^{flowShortage} - v_{n,s,t}^{flowSurplus} \tag{24}$$

$$\forall n \in N^{energy}, s \in S, t \in T:$$

The following constraint applies to storage nodes. Notice that when $t = 1$, we use $v_{n,s,t-1}^{state} = d_n^{initialState}$. Furthermore, note that the terms on the right hand side of Equation (25) are in units of power (MW) and the state of a storage node is in units of energy (MWh). Therefore, the right hand side should be multiplied by the length of the time step. However, in this model the length of each time step is 1 hour and therefore the multiplication is not presented.

$$v_{n,s,t}^{state} - (1 - d_n^{stateLoss})\, v_{n,s,t-1}^{state} = \sum_{(i,j)\in F:\ j=n} v_{i,j,s,t}^{flow} - \sum_{(i,j)\in F:\ i=n} v_{i,j,s,t}^{flow}$$
$$+ d_n^{flowExternal}$$
$$+ v_{n,s,t}^{flowShortage} - v_{n,s,t}^{flowSurplus} \qquad (25)$$
$$\forall n \in N^{storage}, s \in S, t \in T :$$

In the implementation Constraints (24) and (25) are added to the model using function `energy_balance_constraints()`.

5 Examples

5.1 Simple house company

The package includes an example use case which show cases the usage of the package. In this example the house company has a photovoltaic system consisting of solar panels on its roof. In the model, the photovoltaic system is represented by a VRE process that has a capacity of 20 MW. The disposable electricity of the house company is represented by an energy node. The electricity generated by the photovoltaic system feeds into this energy node. The electricity needs of the house company are modeled using a demand time series given as a parameter of the energy node. The energy node is also attached to market node, which represents the electricity grid. The house company can sell its excess energy to the grid and make profit and it can buy energy from the grid when the photovoltaic system does not generate enough for its needs. The system is represented by the graph in Figure 1.

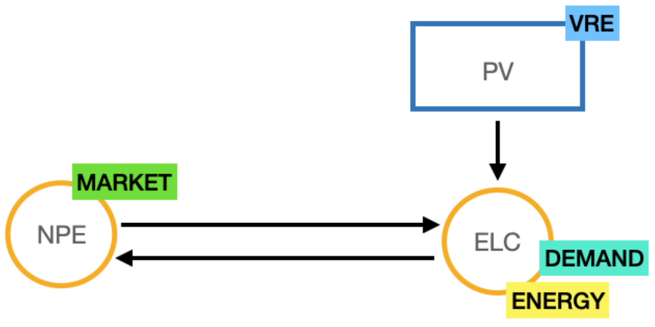

Figure 1: Graphical representation of the simple house company example set up.

In the example, we model two scenarios with four time steps. The first and second scenarios represent a sunny day and a cloudy day respectively. The solar energy availability capacity factor values for each scenario are seen in Table 4. The demand time series is seen in Table 5. In scenario one, the need for electricity is lower because due to the warmth of the sun the heating system is turned down. The market price estimates for the scenarios are seen in Table 6. In the model, the scenario probabilities are 50%.

The model optimises the usage of the photovoltaic system in both of the scenarios so that the costs faced by the house company. Notice that the VOM costs of the photovoltaic system in this example are 0 e/MW. Since this example is very simple, it is always optimal to generate as much energy as possible. This minimises the need to buy energy from the market and if there is excess energy, then that can be sold to the market.

Table 4: CF time series representing solar energy availability.

Scenario	t_1	t_2	t_3	t_4
1	1.0	0.90	0.90	1.0
2	0.4	0.8	0.5	0.6

Table 5: Demand time series representing the house company's need for electricity.

Scenario	t_1	t_2	t_3	t_4
1	-19	-19	-18	-17
2	-23	-24	-22	-22

Table 6: Electricity market price estimates.

Scenario	t_1	t_2	t_3	t_4
1	13	13	13	13
2	13	13	14	12

5.2 Bidding constraint testing example

The bidding constraint example is exactly the same as the simple house company example described above. In this case however, the model is run twice: once exactly as in the previous example and once with the bidding constraints under inspection. The bidding constraints are seen in Equations (26) and (27).

The HouseCompanyEnergyModel.jl provided the ground work to test these constraints, which were then found to behave incorrectly.

$$v^{flow}_{i,n,s,t} - v^{flow}_{n,i,s,t} = v^{flow}_{i,n,s',t} - v^{flow}_{n,i,s',t} \ , \qquad \text{if } c^{energy}_{n,s,t} = c^{energy}_{n,s',t}$$

$$\forall n \in N^{markets}, i \in N : \{(i,n),(n,i)\} \in F, s \in S, s' \in S, t \in T \tag{26}$$

$$v^{flow}_{i,n,s,t} - v^{flow}_{n,i,s,t} \leq v^{flow}_{i,n,s',t} - v^{flow}_{n,i,s',t} \ , \qquad \text{if } c^{energy}_{n,s,t} < c^{energy}_{n,s',t}$$

$$\forall n \in N^{markets}, i \in N : \{(i,n),(n,i)\} \in F, s \in S, s' \in S, t \in T \tag{27}$$

6 Additional comments for future developers

As said in the introduction, this package at the moment only lays the ground work for modeling interesting problems.

Some thoughts that I've had but could not make up my mind about.

- Could the process flow Constraints (12)-(14) and ramping Constraints (16)-(19) only be applied to the output flows of the processes? It seems unnecessarily complicated to declare these constraints for the input and output flows when the efficiency constraint (20) relates these flows to each other.
- Should the process flow Constraints (12) and (14) have the capacity factor upper bound written into them instead of declaring Constraint (15) separately? Would it be motivated to be more concise in this way? Of course this would require still writing out the CF upper bound constraint Constraint (15) out for VRE processes.
- Should transfer flows have a maximum capacity?