Планирование портовых операций, сведение к minizinc-у.

## Основные определения:

**Движущийся объект** – объект, способный перемещаться между локациями – транспортный корабль, буксир или оборудование грузообработки.

**Субъект операции** – движущийся объект, которого перемещают, швартуют/отшвартовывают или с которым выполняют операцию грузообработки.

**Операция перемещения** – у главного объекта меняется локация – операции перемещения между причалами, швартовка и отшвартовка.

## Основные константы:

**n\_intervals** : int – горизонт планирования, количество элементарных временных интервалов.

**n\_operations :** int – количество всех операций.

**n\_locations :** int – количество локаций, чётные локации соответствуют состояниям “отшвартовано”, чётная + 1 – пришвартовано на той физической локации, что и чётная.

**n\_moving\_obj :** int – количество движущихся объектов.

## Описание:

### Подготовка:

Уберём типизацию по основному субъекту операции – заменим каждую операцию на её копии, где вместо типа указан конкретный объект (по количеству подходящих объектов).

### Статус операции:

**op\_status** : array [1..n\_operations, 0..(n\_intervals + 1)] of var bool – статус операции – true – выполняется в текущий интервал, false – не выполняется. В крайних интервалах – 0 и n\_intervals + 1 ничего не выполняется. Основные переменные, остальное так или иначе будет завязано на них.

### Основные параметры-переменные движущегося объекта:

**current\_moving\_operation** : array [1..n\_moving\_obj, 0..(n\_intervals + 1)] of var 0..n\_operations

– номер операции перемещения, в которой участвует объект, или 0, если в таковой операции объект не участвует.

**moving\_op\_of\_obj :** array [1..n\_moving\_obj, 1..n\_operations] of bool – вспомогательные константы,операции перемещения, которые могут затрагивать данный объект – true – могут; false – не могут.

##### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Текущая операция лежит в множестве тех, которые затрагивают данный объект. |
| 2 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 3 |  | (current\_moving\_operation[obj, t] != 0) -> moving\_op\_of\_obj[obj, current\_moving\_operation[obj, t]] |
| 4 |  | ); |
| 5 |  | % Если операция, затрагивающая данный объект в качестве главного, выполняется - то именно она текущая операция перемещения для этого объекта. |
| 6 |  | constraint forall (op in 1..n\_operations, t in 1..n\_intervals where is\_moving\_operation[op]) ( |
| 7 |  | op\_status[op, t] -> (current\_moving\_operation[main\_obj\_of\_operation[op], t] = op) |
| 8 |  | ); |
| 9 |  | % Связь current\_moving\_operation c op\_status. |
| 10 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 11 |  | (current\_moving\_operation[obj, t] != 0) -> (op\_status[current\_moving\_operation[obj, t], t]) |
| 12 |  | ); |

**m\_obj\_loc** : array [1..n\_moving\_obj, 0..(n\_intervals + 1)] of var 1..n\_locations

– локация объекта, в которой он находится, или в которую он перемещается перед началом интервала.

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Направление движения/местоположение объекта может измениться только если перед этим началась операция перемещения, и он не вспомогательный при операции швартовки. |
| 2 |  | constraint forall (obj in 1..n\_moving\_obj, t in 2..(n\_intervals + 1)) ( |
| 3 |  | ((current\_moving\_operation[obj, t - 1] != current\_moving\_operation[obj, t - 2]) /\ |
| 4 |  | (current\_moving\_operation[obj, t - 1] != 0) /\ |
| 5 |  | ((obj = main\_obj\_of\_operation[current\_moving\_operation[obj, t - 1]]) |
| 6 |  | \/ |
| 7 |  | (not is\_mooring\_op[current\_moving\_operation[obj, t - 1]]) |
| 8 |  | ) |
| 9 |  | ) -> (m\_obj\_loc[obj, t] = operations\_destination[current\_moving\_operation[obj, t - 1]]) |
| 10 |  | ); |
| 11 |  | constraint forall (obj in 1..n\_moving\_obj, t in 2..(n\_intervals + 1)) ( |
| 12 |  | ((current\_moving\_operation[obj, t - 1] = current\_moving\_operation[obj, t - 2]) \/ |
| 13 |  | (current\_moving\_operation[obj, t - 1] = 0) \/ |
| 14 |  | ((obj != main\_obj\_of\_operation[current\_moving\_operation[obj, t - 1]]) |
| 15 |  | /\ |
| 16 |  | ( is\_mooring\_op[current\_moving\_operation[obj, t - 1]]) |
| 17 |  | ) |
| 18 |  | ) -> (m\_obj\_loc[obj, t] = m\_obj\_loc[obj, t - 1]) |
| 19 |  | ); |

### Участие объектов в качестве ресурсов, связь с основными переменными:

Основная идея – на каждую операцию, на каждый тип ресурсов, что встречается в ресурсах данной операции (и, разумеется, на каждый интервал) завести счётчик объектов этого типа (**resources\_counter**), которые участвуют как ресурсы(**participation\_as\_resource**) в данной операции.

Если операция выполняется, то все счётчики этой операции должны равняться нужному количеству ресурсов соответствующего типа для её запуска.

Так же обяжем ресурсы не меняться во время всей операции – существенно для поддержания **current\_moving\_operation** при операции перемещения; при операции погрузки – не влияет на наличие решения, может сокращать их множество.

Для простоты реализации все счётчики имеют сквозную нумерацию, информация о них идёт в вспомогательных массивах.

#### Константы:

**operations\_that\_used\_obj\_as\_resource** : array [1..n\_moving\_obj] of set of 1..n\_operations – операции, которые могут использовать данный объект как ресурс.

**is\_moving\_operation** : array [0..n\_operations] of bool – является ли операция операцией перемещения.

**n\_resources\_types** : int - количество различных типов ресурсов.

**objects\_of\_type** : array [1..n\_resources\_types] of set of 1..n\_moving\_obj - все объекты конкретного типа.

**n\_resources\_counters** : int - количество счётчиков ресурсов.

**counter\_type** : array [1..n\_resources\_counters] of 1..n\_resources\_types - тип ресурсов, за которым следит счётчик.

**operation\_of\_counter** : array [1..n\_resources\_counters] of 1..n\_operations - операция, которой принадлежит данный счётчик.

**required\_counter\_values** : array [1..n\_resources\_counters] of int - необходимые значения на счётчиках ресурсов для выполнения операции.

**counters\_of\_operation** : array [1..n\_operations] of set of 1..n\_resources\_counters - счётчики, которые относятся к данной операции.

#### Переменные:

**participation\_as\_resource** : array [1..n\_moving\_obj, 0..(n\_intervals + 1)] of var 0..n\_operations – операция, в которой объект участвует как ресурс, или “0”, если в таковой не участвует.

##### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Граничные значения. |
| 2 |  | constraint forall (obj in 1..n\_moving\_obj) (participation\_as\_resource[obj, 0] = 0); |
| 3 |  | constraint forall (obj in 1..n\_moving\_obj) (participation\_as\_resource[obj, n\_intervals + 1] = 0); |
| 4 |  |  |
| 5 |  | % Только те операции, которые затрагивают данный объект. |
| 6 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 7 |  | (participation\_as\_resource[obj, t] != 0) |
| 8 |  | -> participation\_as\_resource[obj, t] in operations\_that\_used\_obj\_as\_resource[obj] |
| 9 |  | ); |
| 10 |  |  |
| 11 |  | % Связь с текущими операциями перемещения. |
| 12 |  | % Если объект задействован в операции перемещения, которая использует его как ресурс, |
| 13 |  | % то participation\_as\_resource должен указывать на эту операцию. |
| 14 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 15 |  | (current\_moving\_operation[obj, t] in operations\_that\_used\_obj\_as\_resource[obj]) |
| 16 |  | -> |
| 17 |  | (participation\_as\_resource[obj, t] = current\_moving\_operation[obj, t]) |
| 18 |  | ); |
| 19 |  | % Если объект участвует как ресурс в операции перемещения, то это согласованно с current\_moving\_operation. |
| 20 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 21 |  | ((participation\_as\_resource[obj, t] != 0) /\ |
| 22 |  | (is\_moving\_operation[participation\_as\_resource[obj, t]]) |
| 23 |  | ) -> (participation\_as\_resource[obj, t] = current\_moving\_operation[obj, t]) |
| 24 |  | ); |
| 25 |  |  |
| 26 |  | % { Объект участвует где-то в качестве ресурса - соответствующая операция обязана быть активной. } |
| 27 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 28 |  | (participation\_as\_resource[obj, t] != 0) -> op\_status[participation\_as\_resource[obj, t], t] |
| 29 |  | ); |
| 30 |  |  |
| 31 |  | % От начала операции и до конца её ресурсы не могут измениться (в том числе и для погрузки). |
| 32 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..(n\_intervals - 1)) ( |
| 33 |  | ((participation\_as\_resource[obj, t] != 0) /\ |
| 34 |  | (op\_status[participation\_as\_resource[obj, t], t + 1]) |
| 35 |  | ) -> (participation\_as\_resource[obj, t + 1] = participation\_as\_resource[obj, t]) |
| 36 |  | ); |

**resources\_counter** : array [1..n\_resources\_counters , 1..n\_intervals] of var int – счётчик объектов определённого типа, участвующих в операции, к которой относится этот счётчик.

##### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Участие всех необходимых ресурсов в операции. |
| 2 |  | constraint forall (t in 1..n\_intervals, op in 1..n\_operations, counter in counters\_of\_operation[op]) ( |
| 3 |  | op\_status[op, t] -> (resources\_counter[counter, t] = required\_counter\_values[counter]) |
| 4 |  | ); |
| 5 |  | % Определение resources\_counter. |
| 6 |  | constraint forall (counter in 1..n\_resources\_counters, t in 1..n\_intervals) ( |
| 7 |  | resources\_counter[counter, t] = sum (obj in objects\_of\_type[counter\_type[counter]]) ( |
| 8 |  | participation\_as\_resource[obj, t] = operation\_of\_counter[counter] |
| 9 |  | ) |
| 10 |  | ); |

### Наличие на месте и готовность субъектов операции к её началу:

В момент начала операции её субъект должен:

* Быть в нужной локации.
* Если это операция перемещения, то субъект не участвует в других операциях – (для других операций перемещения достигается определением **current\_moving\_operation**).

#### Константы:

**related\_cargo\_op\_using\_obj\_as\_main** : array [1..n\_moving\_obj] of set of 1..n\_operations – операции грузообработки, которые используют объект как главный.

#### Переменные:

**is\_m\_obj\_in\_movement\_before\_start** : array [1..n\_moving\_obj, 0..(n\_intervals + 1)] of var bool – находится ли объект в движении перед началом интервала (в моменте между двумя интервалами). Основная задача – определить, прибыл ли объект в **m\_obj\_loc** или ещё находится в пути.

##### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % В начале никто не движется. |
| 2 |  | constraint forall (obj in 1..n\_moving\_obj) (not is\_m\_obj\_in\_movement\_before\_start[obj, 0]); |
| 3 |  | % Перед первым интервалом все тоже стоят на месте. |
| 4 |  | constraint forall (obj in 1..n\_moving\_obj) (not is\_m\_obj\_in\_movement\_before\_start[obj, 1]); |
| 5 |  |  |
| 6 |  | % Если в предыдущем интервале объект не движется - то в начале этого он не движется, иначе он движется если предыдущая операция не закончилась. |
| 7 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..(n\_intervals + 1)) ( |
| 8 |  | (current\_moving\_operation[obj, t - 1] = 0) |
| 9 |  | -> is\_m\_obj\_in\_movement\_before\_start[obj, t] = false |
| 10 |  | ); |
| 11 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..(n\_intervals + 1)) ( |
| 12 |  | (current\_moving\_operation[obj, t - 1] != 0) |
| 13 |  | -> is\_m\_obj\_in\_movement\_before\_start[obj, t] = (current\_moving\_operation[obj, t - 1] == current\_moving\_operation[obj, t]) |
| 14 |  | ); |

**is\_involved\_in\_cargo\_op** : array [1..n\_moving\_obj, 1..n\_intervals] of var bool – участвует ли объект в этот момент в операции грузообработки.

##### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Определение is\_involved\_in\_cargo\_op. |
| 2 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 3 |  | is\_involved\_in\_cargo\_op[obj, t] = |
| 4 |  | (exists (op in related\_cargo\_op\_using\_obj\_as\_main[obj]) ( |
| 5 |  | op\_status[op, t] |
| 6 |  | )) |
| 7 |  | \/ |
| 8 |  | ((participation\_as\_resource[obj, t] != 0) /\ (not is\_moving\_operation[participation\_as\_resource[obj, t]])) |
| 9 |  | ); |

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Операции перемещения исключают операции грузообработки и наоборот. |
| 2 |  | constraint forall (obj in 1..n\_moving\_obj, t in 1..n\_intervals) ( |
| 3 |  | is\_involved\_in\_cargo\_op[obj, t] -> (current\_moving\_operation[obj, t] = 0) |
| 4 |  | ); |
| 5 |  | % Наличие главных объектов (субъектов) на месте. |
| 6 |  | constraint forall (op in 1..n\_operations, t in 1..n\_intervals, obj = main\_obj\_of\_operation[op]) ( |
| 7 |  | op\_start[op, t] -> ((m\_obj\_loc[obj, t] == main\_obj\_start\_loc[op]) /\ |
| 8 |  | (not is\_m\_obj\_in\_movement\_before\_start[obj, t]) |
| 9 |  | ) |
| 10 |  | ); |

### Наличие на месте и готовность ресурсов операции к её началу:

Идейно – то же самое, что и для субъектов. Заметим, что стартовые локации субъекта и ресурсов в общем случае различаются (например, при отшвартовки).

#### Константы:

**operations\_resources\_start\_loc** : array [1..n\_resources\_counters] of 1..n\_locations – стартовая локация ресурса, необходимая для начала операции.

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | constraint forall ( |
| 2 |  | op in 1..n\_operations, |
| 3 |  | t in 1..n\_intervals, |
| 4 |  | counter in counters\_of\_operation[op], |
| 5 |  | obj in objects\_of\_type[counter\_type[counter]]) ( |
| 6 |  | ((participation\_as\_resource[obj, t] = op) /\ (op\_start[op, t]) |
| 7 |  | ) -> ((m\_obj\_loc[obj, t] == operations\_resources\_start\_loc[counter]) /\ |
| 8 |  | (not is\_m\_obj\_in\_movement\_before\_start[obj, t]) /\ |
| 9 |  | ((not is\_moving\_operation[op]) -> (current\_moving\_operation[obj, t] = 0)) |
| 10 |  | ) |
| 11 |  | ); |

### Непрерывность операций:

Ограничение – для каждого интервала – если операция началась, то она длится нужное количество интервалов, а после – обязательно прерывается.

#### Константы:

**operations\_duration** : array [1..n\_operations] of int - длительность операции в элементарных интервалах.

**is\_continuous\_operation** : array [1..n\_operations] of bool – является ли данная операция непрерывной.

#### Переменные:

**op\_start** : array [1..n\_operations, 0..(n\_intervals + 1)] of var bool – началась ли операция в текущий интервал - *op\_start[i, j] = (op\_status[i, j] /\ not op\_status[i, j - 1]).*

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | constraint forall (i in 1..n\_operations, len = operations\_duration[i] where is\_continuous\_operation[i]) ( |
| 2 |  | (forall (j in 1..(n\_intervals - len + 1)) ( |
| 3 |  | (op\_start[i, j] == 1) -> ( |
| 4 |  | (forall (k in 1..(len - 1)) (op\_status[i, j + k])) |
| 5 |  | /\ |
| 6 |  | (not op\_status[i, j + len]) |
| 7 |  | ) |
| 8 |  | )) /\ |
| 9 |  | (forall (j in (n\_intervals - len + 2)..(n\_intervals + 1)) (op\_start[i, j] == false)) |
| 10 |  | ); |

### Грузообработка:

Необходимо поддержать следующие условия:

* Неотрицательность объёма груза и не превышение максимального размера хранилища.
* Нужные граничные значения.
* Текущий объём груза = предыдущий + величина потока груза в данный интервал + сумма всех изменений объёма от активных операций, связанных с этим хранилищем.

#### Константы:

**n\_cargo\_types** : int – количество типов грузов.

**n\_obj\_with\_storage** : int – количество объектов с хранилищем. Первые *n\_moving\_obj* из них соответствуют движущимся объектам. Если у объекта нет хранилища, то его максимальный объём будет равен нулю.

**max\_storage\_vol** : array [1..n\_obj\_with\_storage] of int – максимальный объём хранилищ.

**initial\_storage\_vol** : array [1..n\_obj\_with\_storage, 1..n\_cargo\_types] of int – начальный объём хранилищ.

**final\_storage\_vol** : array [1..n\_obj\_with\_storage, 1..n\_cargo\_types] of int – конечный объём хранилищ.

**cargo\_flows** : array [1..n\_obj\_with\_storage, 0..(n\_intervals + 1), 1..n\_cargo\_types] of int – потоки грузов.

**n\_loading\_op** : int – количество операций грузообработки.

**loading\_op\_n** : array [1..n\_loading\_op] of 1..n\_operations – номера операций грузообработки среди общего списка операций.

**involved\_operations** : array [1..n\_obj\_with\_storage, 1..n\_cargo\_types] of set of 1..n\_loading\_op – операции, которые изменяют количество этого груза в данном хранилище.

**loading\_op\_delta\_of\_main\_obj** : array [1..n\_operations] of int – изменение хранилища главного объекта операции грузообработки. Если данная операция не грузообработка, тогда здесь и в дальнейшем будут стоять произвольные подходящие значения.

**operations\_main\_stor** : array [1..n\_operations] of 1..n\_obj\_with\_storage – номер хранилища субъекта операции грузообработки.

**operations\_secondary\_stor** : array [1..n\_operations] of 1..n\_obj\_with\_storage – номер другого хранилища (не субъекта) участвующего в операции грузообработки.

**operations\_cargo\_t** : array [1..n\_operations] of 1..n\_cargo\_types – тип груза операции грузообработки.

#### Переменные:

**storage\_used\_volume** : array [1..n\_obj\_with\_storage, 0..(n\_intervals + 1), 1..n\_cargo\_types] of var int – использованный объём хранилища по каждому типу груза на конец интервала.

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Максимальный объём. |
| 2 |  | constraint forall (storage in 1..n\_obj\_with\_storage, t in 0..(n\_intervals + 1)) ( |
| 3 |  | sum (cargo in 1..n\_cargo\_types) (storage\_used\_volume[storage, t, cargo]) <= max\_storage\_vol[storage] |
| 4 |  | ); |
| 5 |  | % Неотрицательность объёма. |
| 6 |  | constraint forall (storage in 1..n\_obj\_with\_storage, t in 0..(n\_intervals + 1), cargo in 1..n\_cargo\_types) ( |
| 7 |  | 0 <= storage\_used\_volume[storage, t, cargo] |
| 8 |  | ); |
| 9 |  | % Начальные значения. |
| 10 |  | constraint forall (storage in 1..n\_obj\_with\_storage, cargo in 1..n\_cargo\_types) ( % Initial values. |
| 11 |  | storage\_used\_volume[storage, 0, cargo] = initial\_storage\_vol[storage, cargo] |
| 12 |  | ); |
| 13 |  | % Конечные значения. |
| 14 |  | constraint forall (storage in 1..n\_obj\_with\_storage, cargo in 1..n\_cargo\_types % Final values. |
| 15 |  | where final\_storage\_vol[storage, cargo] >= 0) ( |
| 16 |  | storage\_used\_volume[storage, n\_intervals + 1, cargo] = final\_storage\_vol[storage, cargo] |
| 17 |  | ); |
| 18 |  | % Изменение грузов в хранилищах. |
| 19 |  | constraint forall (storage in 1..n\_obj\_with\_storage, cargo in 1..n\_cargo\_types, t in 1..(n\_intervals + 1)) ( |
| 20 |  | storage\_used\_volume[storage, t, cargo] = ( |
| 21 |  | storage\_used\_volume[storage, t - 1, cargo] + |
| 22 |  | cargo\_flows[storage, t, cargo] + |
| 23 |  | sum (inv\_op in involved\_operations[storage, cargo]) ( |
| 24 |  | loading\_op\_delta[inv\_op] \* |
| 25 |  | op\_status[loading\_op\_n[inv\_op], t] |
| 26 |  | ) |
| 27 |  | ) |
| 28 |  | ); |

### Ограничение на количество судов, пришвартованных у одного причала:

Будем считать, что на время операции грузообработки не требующей швартовки субъект этой операции условно пришвартован. Так же будем считать условно пришвартованным судно во время операций швартовки и отшвартовки.

Требуется чтобы количество условно либо реально пришвартованных судов у одного причала было не больше одного.

#### Константы:

**twin\_location** : array [1..n\_locations] of 1..n\_locations – локация у того же причала, отличающаяся от данной состоянием швартовки – например, локация “Причал 1, пришвартовано” будет заменена на “Причал 1, отшвартовано” и наоборот.

**related\_unmoored\_cargo\_op** : array [1..n\_locations] of set of 1..n\_operations - операции грузообработки, проводимые без пришвартовки на этой локации.

#### Переменные:

**obj\_in\_loc\_counter** : array [1..n\_locations, 0..(n\_intervals + 1)] of var int – счётчик пришвартованных судов у чётных локаций.

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % Определение obj\_in\_loc\_counter. |
| 2 |  | constraint forall (loc in 1..n\_locations, t in 0..n\_intervals where (loc mod 2) = 0) ( |
| 3 |  | obj\_in\_loc\_counter[loc, t] = (sum (obj in 1..n\_moving\_obj) (m\_obj\_loc[obj, t + 1] = loc)) % Швартовка и нахождение у причала. |
| 4 |  | + (sum (op in related\_unmoored\_cargo\_op[twin\_location[loc]]) (op\_status[op, t])) % Погрузка без швартовки. |
| 5 |  | + (sum (op in 1..n\_operations where (is\_mooring\_op[op] /\ (main\_obj\_start\_loc[op] = loc))) % Отшвартовка. |
| 6 |  | (op\_status[op, t]) |
| 7 |  | ) |
| 8 |  | ); |
| 9 |  | % Само ограничение. |
| 10 |  | constraint forall (loc in 1..n\_locations, t in 0..(n\_intervals + 1) where (loc mod 2) = 0) ( |
| 11 |  | obj\_in\_loc\_counter[loc, t] <= 1 |
| 12 |  | ); |

### Несочетаемые операции:

Из всех несочетаемых операций непокрытыми остались пары из операций грузообработки с общим субъектом, общей локацией и общим хранилищем. Эти пары отдельно вынесены в множество несочетаемых операций с ограничением, не позволяющим им выполняться вместе.

Так же временно в это множество включены пары {перемещение, грузообработка} с общим субъектом и локацией, так как это повышает производительность.

### Окна непогоды:

На всех интервалах непогоды просто ставим условие op\_status[op, t] = false.

### Фиксированные операции:

Некоторые исполнения операций могут быть зафиксированы, требуется их присутствие в ответе в том же виде. Решается обычным фиксированием нужных переменных из op\_status и participation\_as\_resource.

#### Константы:

**n\_fixed\_op** : int – количество фиксированных операций.  
**fixed\_op** : array [1..n\_fixed\_op] of 1..n\_operations – номера фиксированных операций.

**fixed\_op\_resources** : array [1..n\_fixed\_op] of set of 1..n\_moving\_obj – использованные ресурсы.

**fixed\_op\_start** : array [1..n\_fixed\_op] of 1..n\_intervals – первый интервал фиксированной операции.

**fixed\_op\_end** : array [1..n\_fixed\_op] of 1..n\_intervals – последний интервал фиксированной операции.

**is\_fixed** : array [1..n\_operations, 1..n\_intervals] of bool – фиксирована ли операция в данный момент.

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | constraint forall (no in 1..n\_fixed\_op, op = fixed\_op[no]) ( |
| 2 |  | forall (t in fixed\_op\_start[no]..fixed\_op\_end[no]) ( |
| 3 |  | (op\_status[op, t]) /\ |
| 4 |  | forall (obj in fixed\_op\_resources[no]) ( |
| 5 |  | participation\_as\_resource[obj, t] = op |
| 6 |  | ) |
| 7 |  | ) |
| 8 |  | ); |

### Критерий оптимизации:

Предыдущие ограничения описывают все логические правила, которым должно удовлетворять решение. Осталось только ввести критерий оптимальности решения – назовём решение оптимальным, если последний интервал времени, в который выполнялась хотя бы одна операция, был наименьшим среди всех возможных решений.

#### Переменные:

**is\_not\_terminated** : array [1..(n\_intervals + 1)] of var bool – означает не закончились ли все операции до текущего интервала.

#### Реализация:

|  |  |  |
| --- | --- | --- |
| 1 |  | % В конце всё остановится. |
| 2 |  | constraint (is\_not\_terminated[n\_intervals + 1] == false); |
| 3 |  | % Действия ещё не остановились, если в текущий интервал что-нибудь происходит или они не остановятся в следующий интервал. |
| 4 |  | constraint forall (t in 1..n\_intervals) ( |
| 5 |  | is\_not\_terminated[t] == ( |
| 6 |  | (exists (op in 1..n\_operations) (op\_status[op, t])) |
| 7 |  | \/ |
| 8 |  | is\_not\_terminated[t + 1] |
| 9 |  | ) |
| 10 |  | ); |
| 11 |  | solve minimize sum(is\_not\_terminated); |

### Оптимизация “сдвиг в начало” (только идея):

Основная идея – если операция может быть сдвинута на один интервал к началу без изменения других операций, то она должна быть сдвинута.

Рассмотрим операцию длительностью в один интервал. Очевидно, что она может быть выполнена в данный интервал, если:

1. Не мешают окна непогоды.
2. Главный объект на месте и не участвует в операции перемещения.
3. Достаточно свободных ресурсов на нужном месте.
4. Если это операция перемещения, то субъект не участвует в операциях погрузки.
5. Если это операция пришвартовки, то причал должен быть свободен.
6. Если это операция грузообработки, то она не выведет за границы дозволенного (меньше нуля или выше максимального объёма).
7. Если это операция грузообработки **без швартовки**, то причал должен быть свободен.
8. Не выполняется ни одной конфликтующей операции.

И сама оптимизация – если операция не фиксирована, то её невозможно сдвинуть на один интервал (в предыдущем (от старта) интервале невозможно выполнить эту операцию, считая, что её длительность равна одному кванту времени).