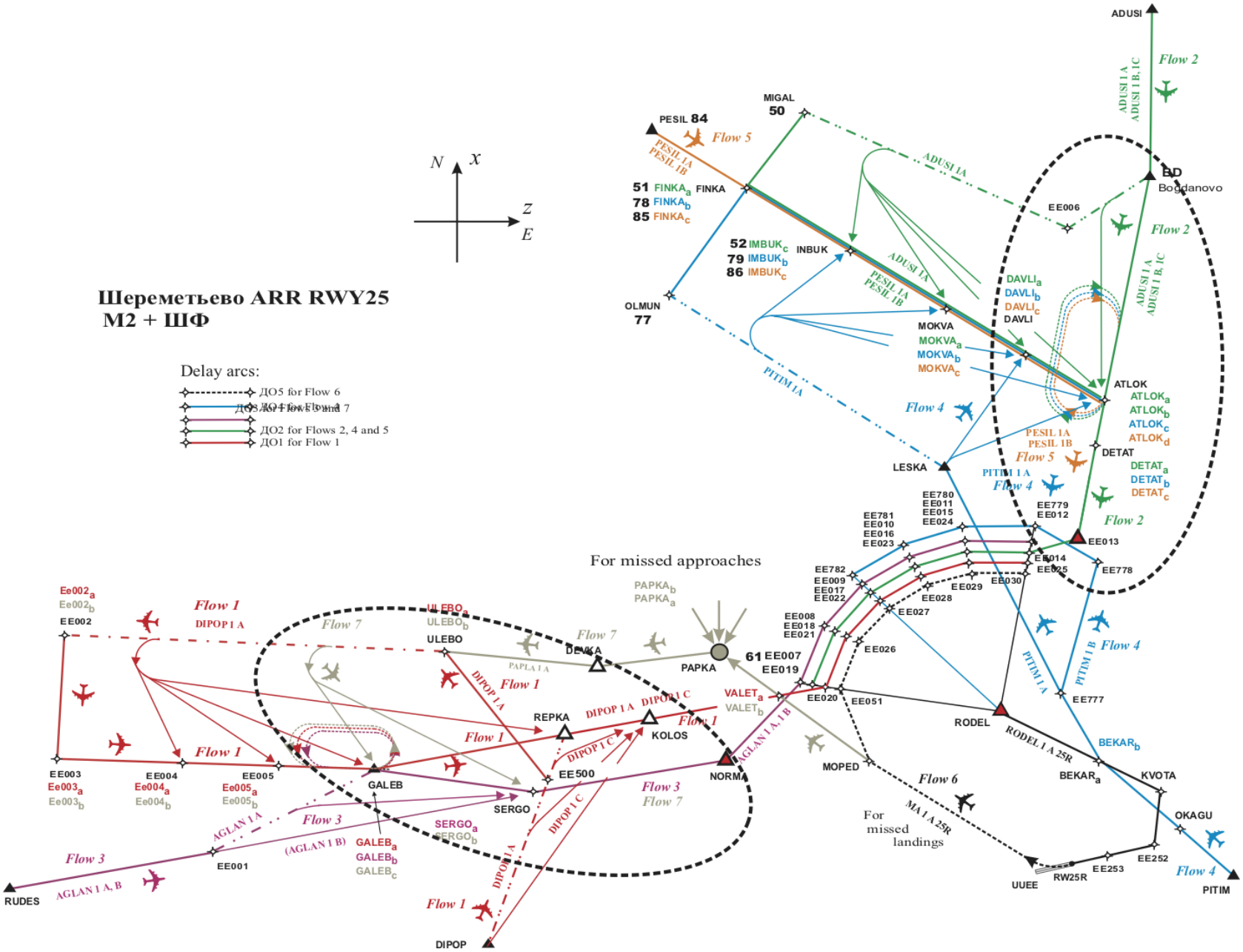
Процедура оптимизации слитой очереди воздушных судов с учётом их типов

А.А. Спиридонов

18 мая 2020 г.


Шереметьево ARR RWY25
М2 + ШФ
Delay arcs:
ДО5 for Flow 6
ДО4 for Flow 3
ДО3 for Flows 3 and 7
ДО2 for Flows 2, 4 and 5
ДО1 for Flow 1
N
X
Z
E
For missed approaches
ADUSI
Flow 2
ADUSI 1A
ADUSI 1B, 1C
MIGAL
50
PESIL 84
Flow 5
PESIL 1A
PESIL 1B
51 FINKA$_a$ FINKA
78 FINKA$_b$
85 FINKA$_c$
52 IMBUK$_c$ INBUK
79 IMBUK$_b$
86 IMBUK$_c$
OLMUN
77
PITIM 1A
BD
Bogdanovo
EE006
DAVLI$_a$
DAVLI$_b$
DAVLI$_c$
DAVLI
MOKVA
MOKVA$_a$
MOKVA$_b$
MOKVA$_c$
ATLOK
ATLOK$_a$
ATLOK$_b$
ATLOK$_c$
ATLOK$_d$
DETAT
DETAT$_a$
DETAT$_b$
DETAT$_c$
LESKA
Flow 4
Flow 5
PITIM 1A
PESIL 1A
PESIL 1B
EE780
EE011
EE015
EE024
EE779
EE012
EE781
EE010
EE016
EE023
EE013
EE014
EE025
EE778
EE782
EE009
EE017
EE022
EE030
EE029
EE028
EE027
EE008
EE018
EE021
EE026
EE020
EE051
RODEL
EE777
PITIM 1B
BEKAR$_b$
BEKAR$_a$
KVOTA
OKAGU
PITIM
EE252
EE253
RW25R
UUEE
RODEL 1A 25R
MA 1A 25R
MOPED
Flow 6
For missed landings
61 EE007
EE019
PAPKA$_b$
PAPKA$_a$
PAPKA
VALET$_a$
VALET$_b$
AGLAN 1A, 1B
NORMA
Ee002$_a$
Ee002$_b$
EE002
DIPOP 1A
Flow 1
Flow 7
ULEBO$_a$
ULEBO$_b$
ULEBO
DEVKA
PAPLA 1A
REPKA
DIPOP 1C
KOLOS
EE003
Ee003$_a$
Ee003$_b$
EE004
Ee004$_a$
Ee004$_b$
EE005
Ee005$_a$
Ee005$_b$
GALEB
EE500
SERGO
SERGO$_a$
SERGO$_b$
Flow 3
AGLAN 1A
(AGLAN 1B)
GALEB$_a$
GALEB$_b$
GALEB$_c$
EE001
AGLAN 1A, B
RUDES
DIPOP

Постановка задачи

- $\{t_i^{\text{nom}}\}_{i=1}^{N}$ — упорядоченный по возрастанию набор номинальных моментов прибытия воздушных судов (ВС) в точку слияния;
- $t_i^{\text{nom}} \to [t_i^{\text{nom}} - t_i^{\text{acc}}, t_i^{\text{nom}} + t_i^{\text{dec}}]$ — интервал варьирования изменённого момента прибытия каждого ВС;
- $\tau_{i,j}^{\text{safe}}$ — минимальный безопасный временной интервал между i-м и j-м ВС в результирующей очереди.

Постановка задачи

Таким образом, возникает оптимизационная задача

$$F(\{t_i\}, \{t_i^{\text{nom}}\}) = \sum_{i=1}^{N} f(t_i, t_i^{\text{nom}}) \to \min$$

при ограничениях

$$t_i \in [t_i^{\text{nom}} - t_i^{\text{acc}}, t_i^{\text{nom}} + t_i^{\text{dec}}],$$
$$\forall\, 1 \leqslant i < j \leqslant N \quad t_j - t_i \geqslant \tau_{i,j}^{\text{safe}}$$

Постановка задачи

Таким образом, возникает оптимизационная задача

$$F(\{t_i\}, \{t_i^{\text{nom}}\}) = \sum_{i=1}^{N} f(t_i, t_i^{\text{nom}}) \to \min$$

при ограничениях

$$t_i \in [t_i^{\text{nom}} - t_i^{\text{acc}}, t_i^{\text{nom}} + t_i^{\text{dec}}],$$
$$\forall\, 1 \leqslant i < j \leqslant N \quad t_j - t_i \geqslant \tau_{i,j}^{\text{safe}}$$

В случае разнотиповых судов важен их порядок!

Нелинейные критерии

На предыдущем этапе работы рассматривались нелинейные критерии с жёсткими и мягкими ограничениями, но без учёта типов ВС.

Нелинейные критерии

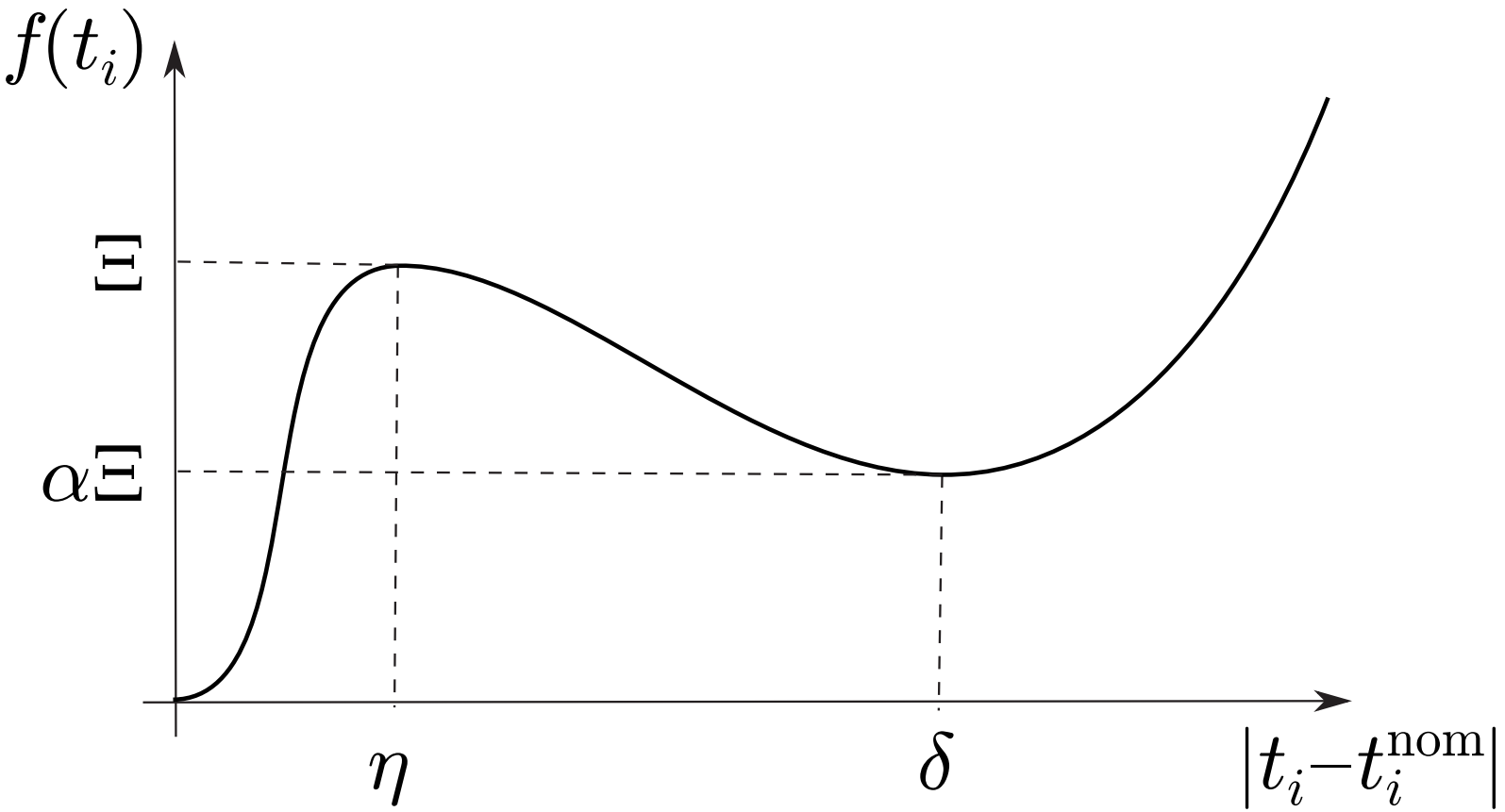
На предыдущем этапе работы рассматривались нелинейные критерии с жёсткими и мягкими ограничениями, но без учёта типов ВС.

Например, квадратичный критерий:

$$F(\{t_i\}, \{t_i^{\text{nom}}\}) = \sum_{i=1}^{N} (t_i - t_i^{\text{nom}})^2 \to \min$$

Нелинейные критерии

Критерий с ограничением на минимальную величину вариации:



Нелинейные критерии

В случае выпуклости критерия $f(t_i, t_i^{\text{nom}})$ для одного суда выпуклым является и суммарный критерий $F(\{t_i\}, \{t_i^{\text{nom}}\})$. Однако если критерий f одного судна невыпуклый, то суммарный критерий F становится многоэкстремальным и плохо поддаётся минимизации.

Нелинейные критерии

В случае выпуклости критерия $f(t_i, t_i^{\text{nom}})$ для одного суда выпуклым является и суммарный критерий $F(\{t_i\}, \{t_i^{\text{nom}}\})$. Однако если критерий f одного судна невыпуклый, то суммарный критерий F становится многоэкстремальным и плохо поддаётся минимизации.

Поэтому вернёмся обратно к простейшему кусочно-линейному критерию в виде суммы $f(t_i, t_i^{\text{nom}}) = |t_i - t_i^{\text{nom}}|$, но с учётом разных типов ВС.

Учёт порядка ВС. Обозначения

Рассматривались три типа судов: «лёгкий» (L, light), «средний» (M, medium), «тяжёлый» (H, heavy).

σ_i — тип i -го ВС, σ выбирается из множества $\{\mathsf{L}, \mathsf{M}, \mathsf{H}\}$.

Промежуток безопасности $\tau^{\mathsf{safe}}_{\sigma_i,\sigma_j}$, (с)	лёгкий	средний	тяжёлый
лёгкий	60	60	60
средний	180	60	60
тяжёлый	180	120	120

Величины варьирования момента прибытия t_i^{acc}, t_i^{dec} также зависят от типа ВС, однако сейчас в моделировании используются одинаковые для всех типов значения $t_i^{\mathsf{acc}} = t^{\mathsf{acc}} = 8$ минут, $t_i^{\mathsf{dec}} = t^{\mathsf{dec}} = 22$ минуты.

Идеи обработки ансамбля ВС

Количество всевозможных перестановок N судов — $N!$, при $N \geqslant 13$ очень большая величина. Нужно ограничить перебор перестановок.

В расчётах хочется пользоваться вычислительными возможностями обыкновенного персонального компьютера, поскольку данные алгоритмы, скорее всего, будут исполняться на рабочем месте диспетчера УВД.

Идеи обработки ансамбля ВС

Основная идея — разбиение исходной группы на подгруппы меньшего размера (кластеризация), оптимизировать которые можно независимо. Выделение таких подгрупп осуществляется построением расписания для судов в их исходном порядке, а затем обнаружением последовательностей судов, разделенных промежутком, большим определенного порога. Априорная разумная величина такого промежутка — 5 минут.

Идеи обработки ансамбля ВС

Основная идея — разбиение исходной группы на подгруппы меньшего размера (кластеризация), оптимизировать которые можно независимо. Выделение таких подгрупп осуществляется построением расписания для судов в их исходном порядке, а затем обнаружением последовательностей судов, разделенных промежутком, большим определенного порога. Априорная разумная величина такого промежутка — 5 минут.

Вторая идея — ограничение изменения позиции ВС по отношению к исходному порядку не более чем на оговоренную величину. Это ограничение в разных экспериментах бралось равным 4 и 5.

Идеи обработки ансамбля ВС

Третья идея — использование разных промежутков варьирования для разных ВС. В численных экспериментах с одинаковым максимальным промежутком варьирования момента прибытия (8 мин ускорения, 22 мин замедления) величина этого промежутка такова, что весьма слабо сокращает перебор. Но в ситуации с судами, имеющими различные промежутки варьирования, эта идея может дать выигрыш в производительности.

Алгоритм генерации перестановки группы ВС

Идея о перестановке судов в очереди при учёте лишь величины отклонений назначенного момента прибытия от номинального, вообще говоря, **не даёт более оптимального решения**.

Пример: (H,1500), (L,1560), (H,1620).
Имеется конфликт между 1-м и 2-м ВС. Разрешить его можно или ускорив 1-е судно на 120 сек, либо ускорив 2-е судно на 120 сек. С точки зрения функции штрафа оба варианта равноценны, но по требованиям УВД в таких ситуациях требуется маневрировать лёгким судном.

Алгоритм генерации перестановки группы ВС

Отразить приоритет манёвров более лёгких судов можно введя зависимость функции штрафа от типа ВС:

$$f(t_i, t_i^{\text{nom}}, \sigma_i) = \beta_{\sigma_i} \cdot |t_i - t_i^{\text{nom}}|.$$

В дальнейших экспериментах считаем $\beta_{\text{L}} = 1.0$, $\beta_{\text{M}} = 3.0$, $\beta_{\text{H}} = 5.0$.

Идея перебора перестановок ВС

Основной критерий — «плотность» очереди: сумма интервалов безопасности между всеми парами соседних ВС в очереди.

Начиная с исходного порядка судов (в соответствии с номинальными моментами прибытия) будем менять местами подгруппы ВС так, чтобы очередь становилась более плотной, то есть сумма интервалов безопасности **строго** уменьшалась.

В принципе, рассматривался и вариант, когда исследовались перестановки, которые не ухудшают плотность очереди. Однако в этом случае перебор становился слишком большим.

Типы перестановок

Рассмотрим последовательность судов:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Типы перестановок

Рассмотрим последовательность судов:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Выделим в ней две подряд идущих подгруппы:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Типы перестановок

Рассмотрим последовательность судов:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Выделим в ней две подряд идущих подгруппы:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Попробуем переставить их местами:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_{j+1}\, \ldots\, v_k\, v_i\, \ldots\, v_j\, v_{k+1}\, \ldots\, v_N$$

Типы перестановок

Рассмотрим последовательность судов:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Выделим в ней две подряд идущих подгруппы:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_i\, \ldots\, v_j\, v_{j+1}\, \ldots\, v_k\, v_{k+1}\, \ldots\, v_N$$

Попробуем переставить их местами:

$$v_1\, v_2\, \ldots\, v_{i-1}\, v_{j+1}\, \ldots\, v_k\, v_i\, \ldots\, v_j\, v_{k+1}\, \ldots\, v_N$$

Новая перестановка лучше, если она более плотная.

Типы перестановок

Запишем формально, что новая перестановка более плотная.

$$\ldots \sigma_1 \, [\sigma_2 \ldots \sigma_3] \, [\sigma_4 \ldots \sigma_5] \, \sigma_6 \ldots \quad \longrightarrow$$
$$\longrightarrow \quad \ldots \sigma_1 \, [\sigma_4 \ldots \sigma_5] \, [\sigma_2 \ldots \sigma_3] \, \sigma_6 \ldots$$

Сравнением величины $s^{\mathsf{mid}}_{\mathsf{old}} = \tau^{\mathsf{safe}}_{\sigma_1,\sigma_2} + \tau^{\mathsf{safe}}_{\sigma_3,\sigma_4} + \tau^{\mathsf{safe}}_{\sigma_5,\sigma_6}$ и $s^{\mathsf{mid}}_{\mathsf{new}} = \tau^{\mathsf{safe}}_{\sigma_1,\sigma_4} + \tau^{\mathsf{safe}}_{\sigma_5,\sigma_2} + \tau^{\mathsf{safe}}_{\sigma_3,\sigma_6}$

Перебираем $\sigma_1 \in \{\mathsf{L}, \mathsf{M}, \mathsf{H}\}$, $\sigma_2 \in \{\mathsf{L}, \mathsf{M}, \mathsf{H}\}$, …, $\sigma_6 \in \{\mathsf{L}, \mathsf{M}, \mathsf{H}\}$ и выбираем те их комбинации, для которых $s^{\mathsf{mid}}_{\mathsf{new}} < s^{\mathsf{mid}}_{\mathsf{old}}$.

Перестановок в середине очереди — 162 шт.

Типы перестановок

Аналогично:

Перестановок в начало очереди — 54 шт.

$[\sigma_2 \dots \sigma_3]\,[\sigma_4 \dots \sigma_5]\,\sigma_6 \dots \quad \longrightarrow \quad [\sigma_4 \dots \sigma_5]\,[\sigma_2 \dots \sigma_3]\,\sigma_6 \dots$

Сравнение величин $s^{\text{beg}}_{\text{old}} = \tau^{\text{safe}}_{\sigma_3,\sigma_4} + \tau^{\text{safe}}_{\sigma_5,\sigma_6}$ и $s^{\text{beg}}_{\text{new}} = \tau^{\text{safe}}_{\sigma_5,\sigma_2} + \tau^{\text{safe}}_{\sigma_3,\sigma_6}$

Перестановка в конец очереди — 68 шт.

$\dots \sigma_1\,[\sigma_2 \dots \sigma_3]\,[\sigma_4 \dots \sigma_5] \quad \longrightarrow \quad \dots \sigma_1\,[\sigma_4 \dots \sigma_5]\,[\sigma_2 \dots \sigma_3]$

Сравнение величин $s^{\text{end}}_{\text{old}} = \tau^{\text{safe}}_{\sigma_1,\sigma_2} + \tau^{\text{safe}}_{\sigma_3,\sigma_4}$ и $s^{\text{end}}_{\text{new}} = \tau^{\text{safe}}_{\sigma_1,\sigma_4} + \tau^{\text{safe}}_{\sigma_5,\sigma_2}$

Схема алгоритма

1. Кластеризация исходного набора ВС;
2. Независимая оптимизация каждого кластера
    1. Перебор шаблонов перестановок;
    2. Поиск очередного вхождения текущего шаблона;
    3. Осуществление соответствующей перестановки и вычисление основного критерия на полученном порядке;
    4. Запоминание полученного порядка для проведения перестановок в нём.

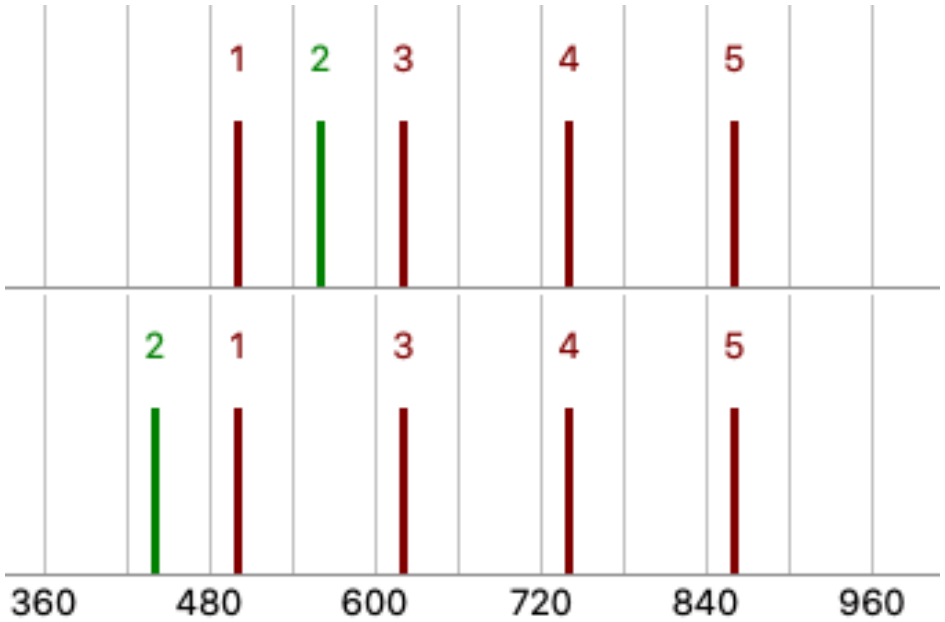
Схема алгоритма

1. Кластеризация исходного набора ВС;
2. Независимая оптимизация каждого кластера
   1. Перебор шаблонов перестановок;
   2. Поиск очередного вхождения текущего шаблона;
   3. Осуществление соответствующей перестановки и вычисление основного критерия на полученном порядке;
   4. Запоминание полученного порядка для проведения перестановок в нём.

Графовая формализация: узел — порядок судов, из одного узла в другой идёт дуга, если в первом порядке есть улучшающая перестановка, дающая второй порядок. Перебор узлов DAG'а и поиск узла, соотвествующего порядку, дающему минимальное значение функционала. Поиск в ширину.

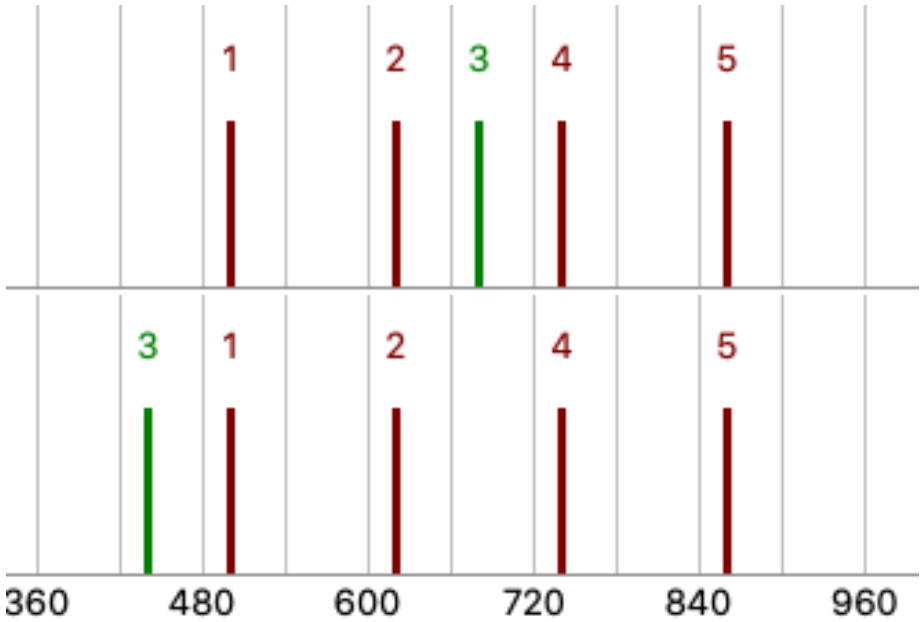
Примеры работы алгоритма

t_i^{nom}	500	560	620	740	860
Тип	H	L	H	H	H
t_i	500	440	620	740	860



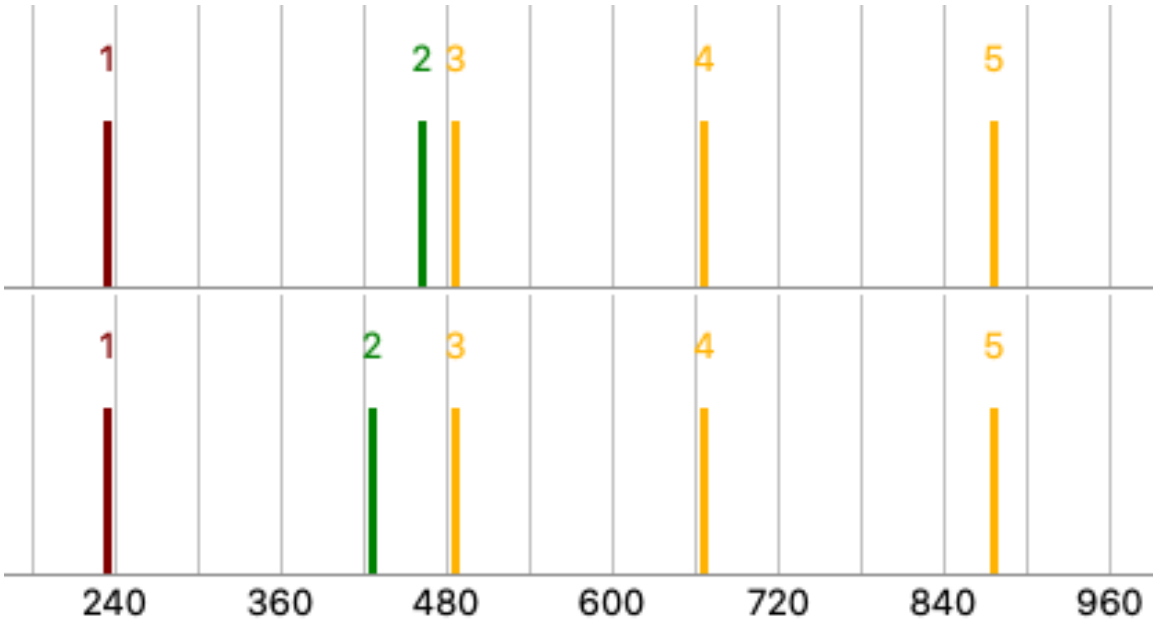
Примеры работы алгоритма

t_i^{nom}	500	620	680	740	860
Тип	H	H	L	H	H
t_i	500	620	440	740	860



Примеры работы алгоритма

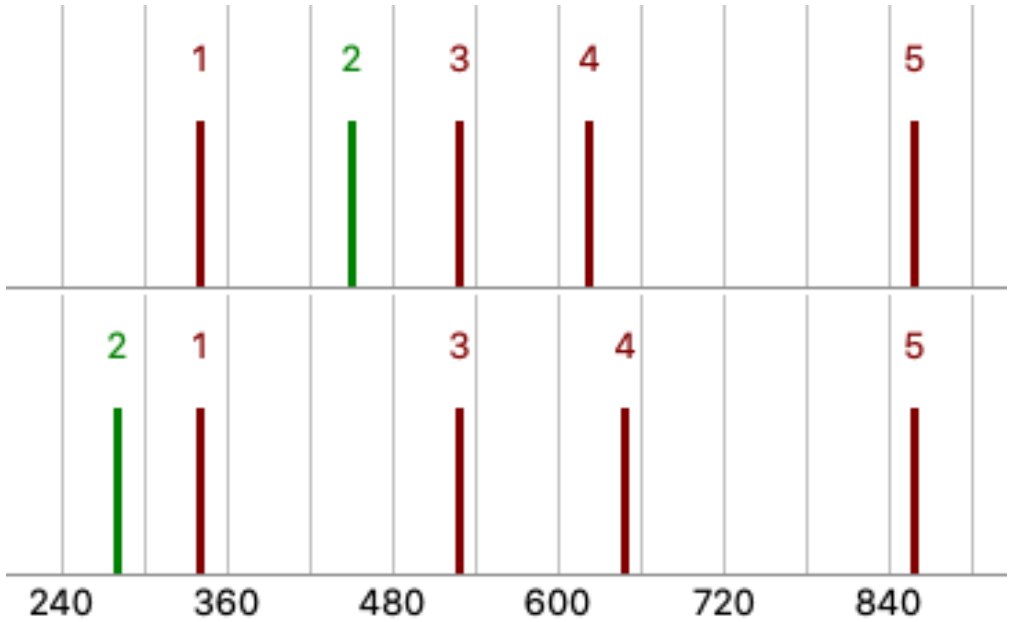
t_i^{nom}	235.6	462.9	487.5	666.3	877.4
Тип	H	L	M	M	M
t_i	235.6	427.5	487.5	666.3	877.4



Требуемый интервал получен только манёвром лёгкого судна.

Примеры работы алгоритма

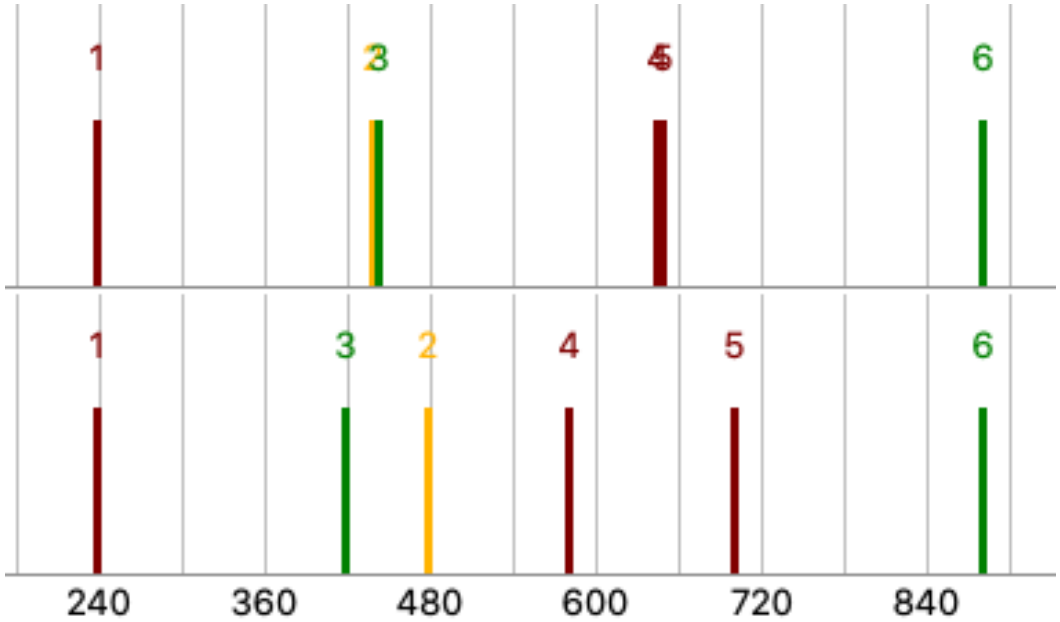
t_i^{nom}	340.3	451.2	529.4	623.9	859.0
Тип	H	L	H	H	H
t_i	340.3	280.3	529.4	649.4	859.0



Также необходим манёвр тяжёлого судна с номером 4.

Примеры работы алгоритма

t_i^{nom}	238.6	439.0	442.7	644.4	649.2	881.6
Тип	H	M	L	H	H	L
t_i	238.6	478.6	418.6	581.6	701.6	881.6

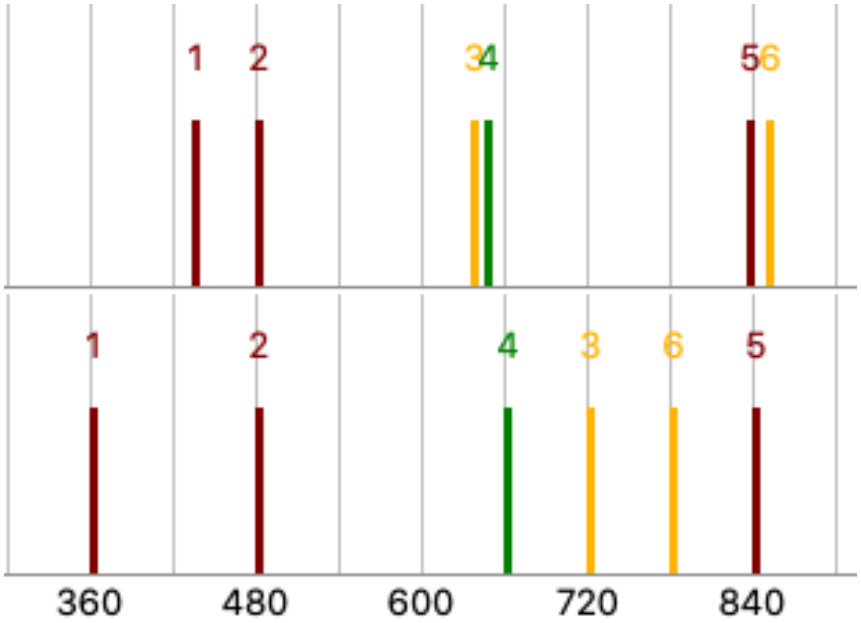


Изменён порядок 2-го и 3-го судов.

В паре 4-го и 5-го судов можно двигать любое или оба, но если двигать одно — возникает наведённый конфликт.

Примеры работы алгоритма

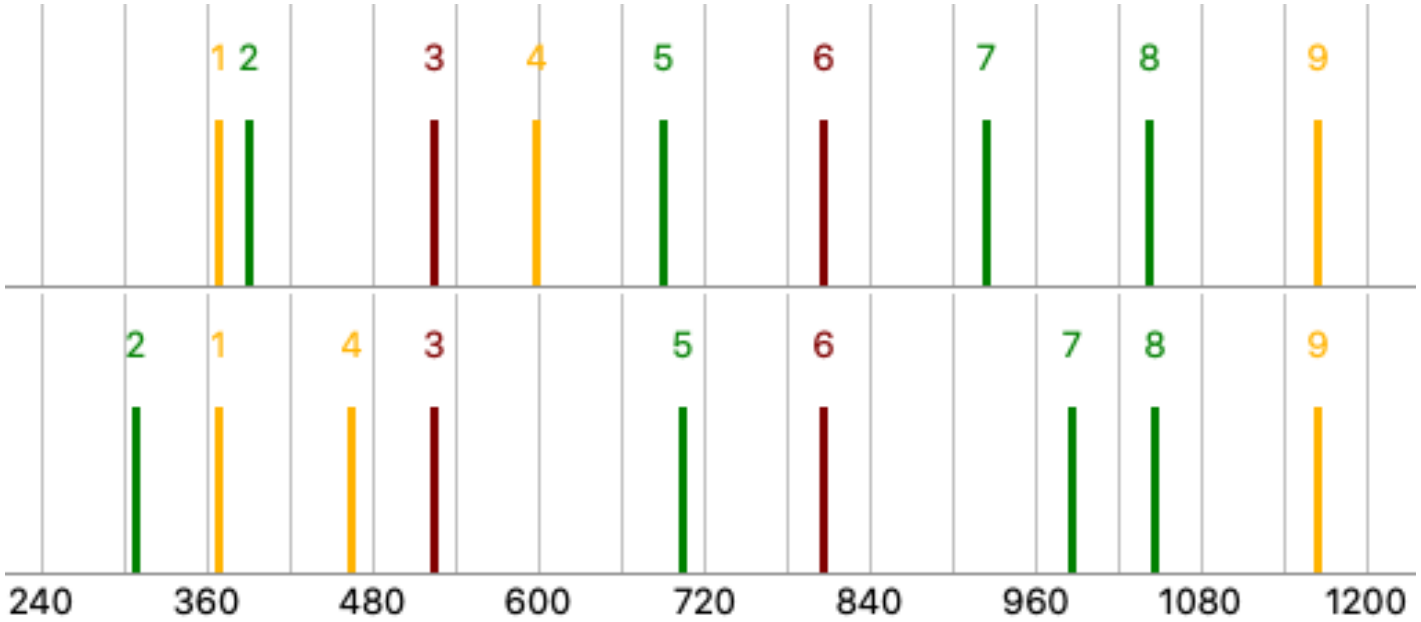
t_i^{nom}	436.9	483.4	639.6	649.0	838.0	852.7
Тип	H	H	M	L	H	M
t_i	363.4	483.4	723.4	663.4	843.4	783.4



По удачному стечению обстоятельств удалось вписать средние суда между лёгким и тяжёлым без изменения их момента прибытия.

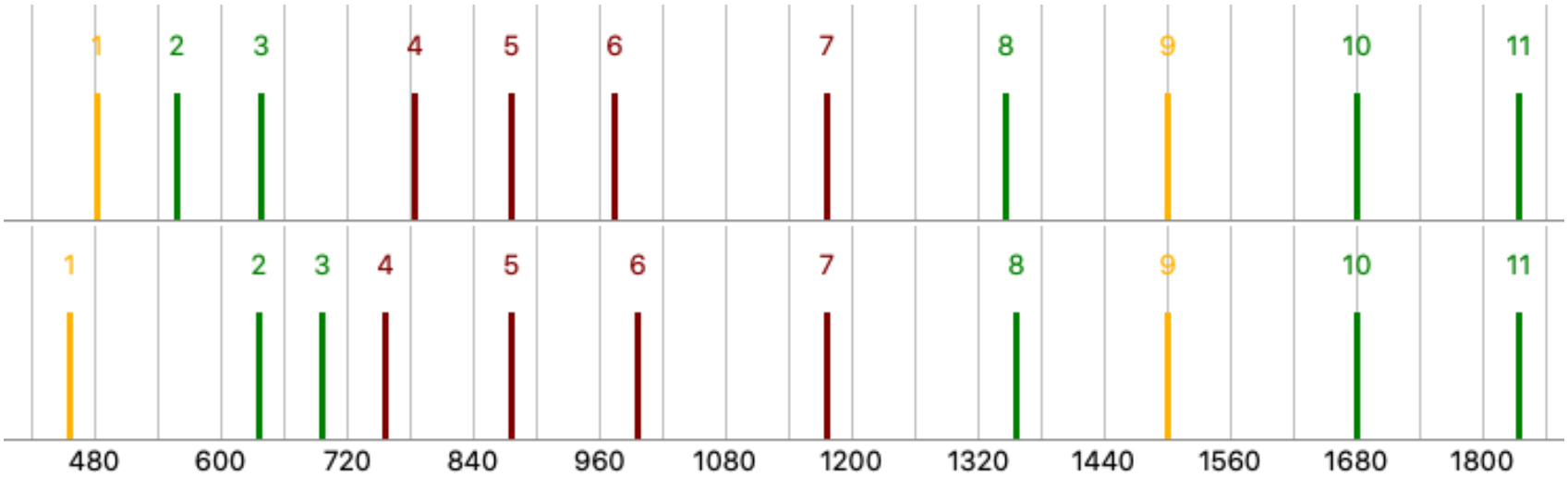
Примеры работы алгоритма

t_i^{nom}	369.2	391.9	524.6	599.0	690.4	806.6	925.1	1042.5	1164.3
Тип	M	L	H	M	L	H	L	L	M
t_i	369.2	309.2	524.6	464.6	704.6	806.6	986.6	1046.6	1164.3



Примеры работы алгоритма

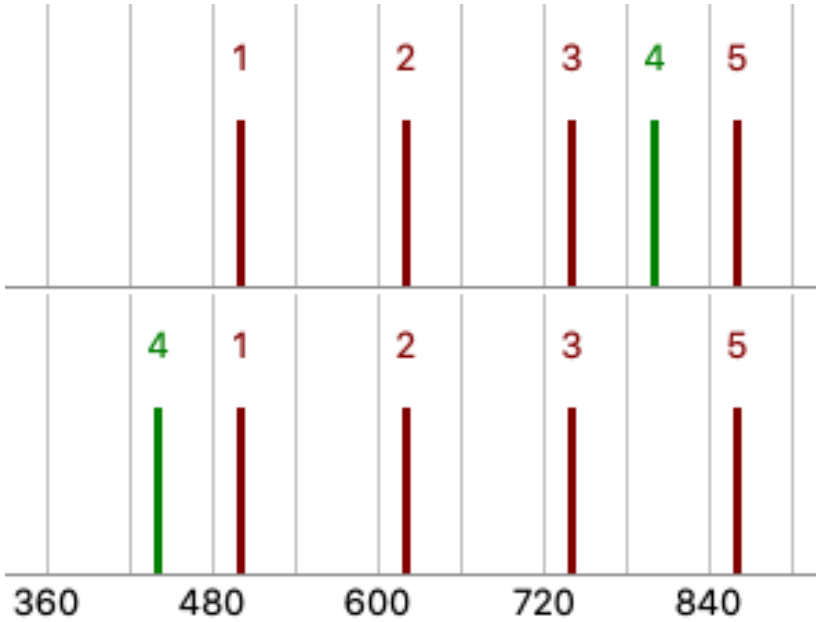
t_i^{nom}	483.8	558.7	638.6	785.1	877.8	975.4	1176.0	1346.2	1500.1	1680.6	1835.6
Тип	M	L	L	H	H	H	H	L	M	L	L
t_i	457.8	637.8	697.8	757.8	877.8	997.8	1176.0	1356.0	1500.1	1680.6	1835.6



Возникла необходимость манёвра тяжёлых судов.

Примеры работы алгоритма

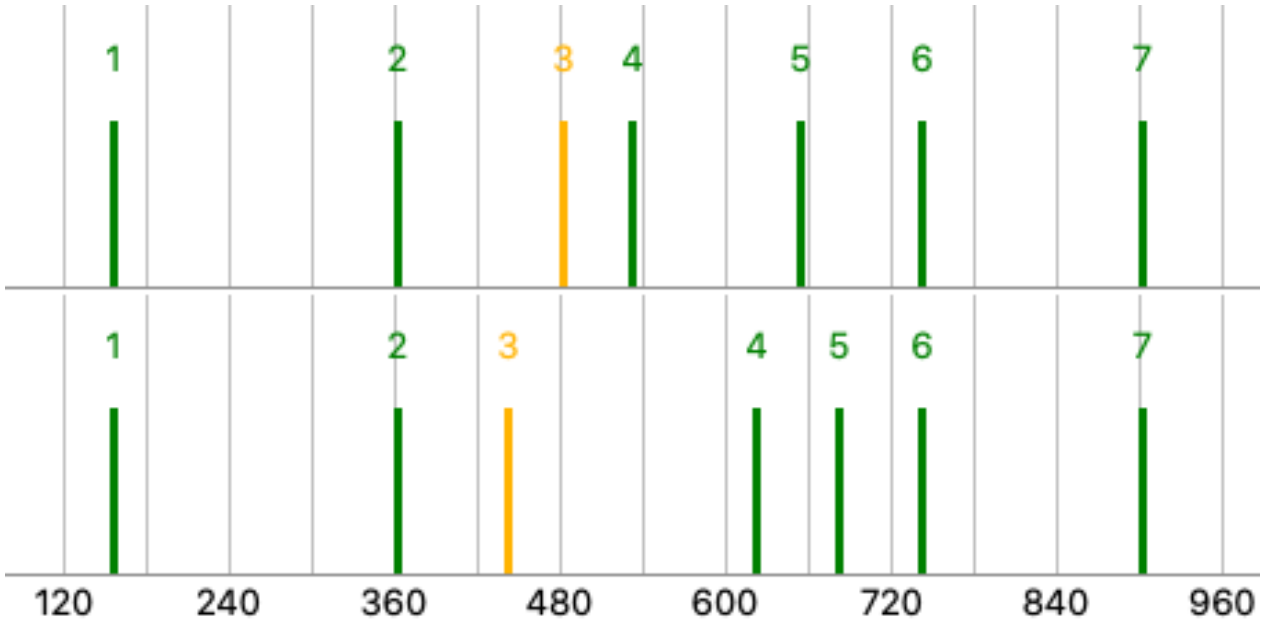
t_i^{nom}	500	620	740	800	860
Тип	H	H	H	L	H
t_i	500	620	740	440	860



Было бы логичнее задержать лёгкое судно на 2 минуты, однако с точки зрения алгоритма такая перестановка менее оптимальная, поскольку ухудшает плотность очереди.

Примеры работы алгоритма

t_i^{nom}	156.5	363.7	482.7	532.7	655.3	742.8	903.3
Тип	L	L	M	L	L	L	L
t_i	156.5	363.7	442.8	622.8	682.8	742.8	903.3



Производительность алгоритма. Среднее время

Численные эксперименты проводились на компьютере с процессором 2.2GHz Intel Core i7-9100 (6 ядер с Hyper-Threading) и памятью 16GB 2300MHz DDR4. Счёт проводился в однопоточном режиме на одном ядре процессора. В таблице приведено среднее время работы алгоритма в миллисекундах для 10000 тестов:

Количество ВС / Огран-е на изменение позиции	5	6	7	8	9	10	11	12	13
4	0.3	1.6	6.0	17.0	46.0	112.5	314.2	747.2	1919.0
5	0.3	2.1	9.1	33.2	115.6	369.7	1001.0	3246.7	9704.0

Производительность алгоритма. Худшее время

Наибольшее время работы алгоритма среди сгенерированных ансамблей в секундах:

Огран-е на изменение позиции \ Количество ВС	5	6	7	8	9	10	11	12	13
4	0.007	0.027	0.109	0.5	1.5	6.2	24.5	78.4	253.4
5	0.006	0.036	0.161	1.0	5.0	21.6	37.4	270.2	740.6

Заключение

Разработана процедура генерации оптимального порядка в очереди ВС разных типов при оптимизации критерия в виде суммы модулей отклонений с весовыми коэффициентами, оценена эффективность её работы.

Кроме того, в некоторой степени, данная процедура также уменьшает количество судов, чьи моменты прибытия были изменены, однако этот аспект работы процедуры может быть улучшен.

Заключение

Полученная статистика показывает, что, как и ожидалось, время счёта растёт экспоненциально по количеству ВС в обрабатываемом ансамбле. Однако, среднее время остаётся приемлемым даже для 13 ВС, может быть и для 14. Требуется сбор более полной статистики: распределение по длительностям счёта, среднее количество кластеров в ансамбле (сейчас эта информация не собиралась), возможно, что-нибудь ещё.

В настоящее время ведётся разработка параллельной версии поиска в ширину для ускорения вычислений с целью получения подобной статистики за разумное время.