

Тема 10. Алгоритмы комбинаторной оптимизации

Задачи комбинаторной оптимизации

Пример 1. Производство краски

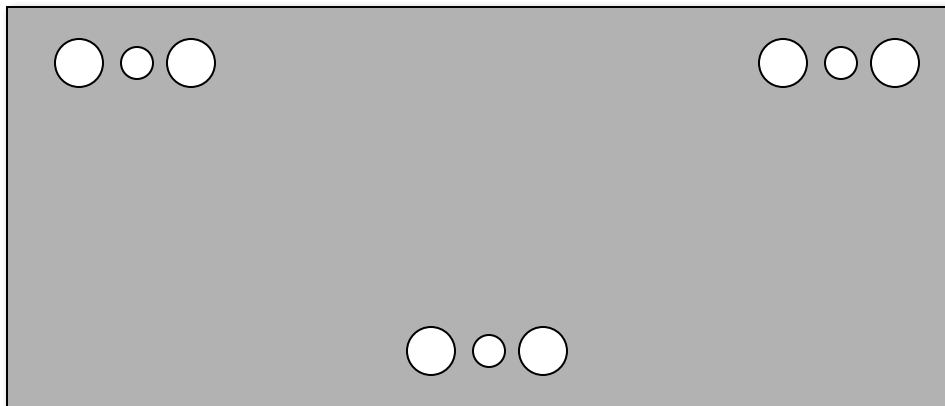


В производстве краски при смене цвета требуется очистка оборудования. Время очистки зависит от того, с какой краски на какую осуществляется переход. Требуется выбрать последовательность смены цвета, которая давала бы минимальную длительность производственного цикла.



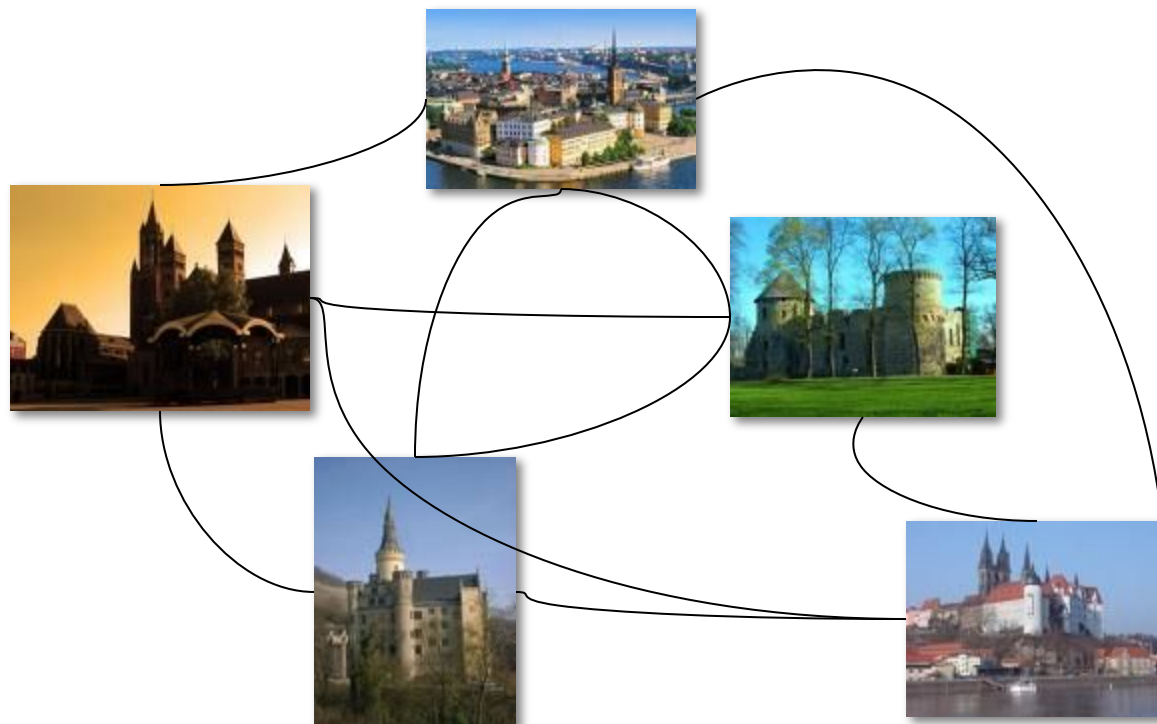
Задачи комбинаторной оптимизации

Пример 2. Координатно-пробивной пресс со сменным инструментом



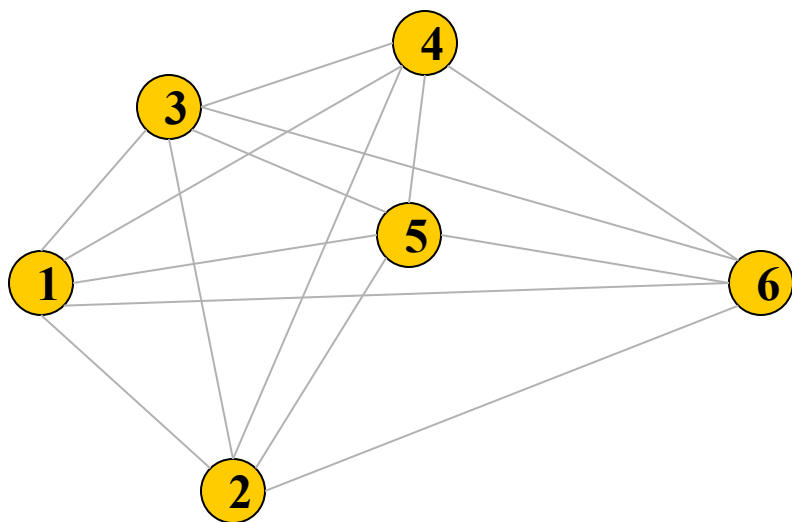
При изготовлении деталей на координатно-пробивном прессе время производственного цикла включает время перемещения заготовки и время смены инструмента. Требуется выбрать последовательность пробивки отверстий, которая давала бы минимальную длительность производственного цикла.

Задача коммивояжёра



Требуется объехать все города, побывав в каждом только один раз, и затратив минимальное время на весь путь.

Задача коммивояжёра. Структура данных



C_{ij}	1	2	3	4	5	6
1		6	4	8	7	14
2	6		7	11	7	10
3	4	7		4	3	10
4	8	11	4		5	11
5	7	7	3	5		7
6	14	10	10	11	7	

C_{ij} = расстояние от i до j

$$C_{ij} = C_{ji}$$

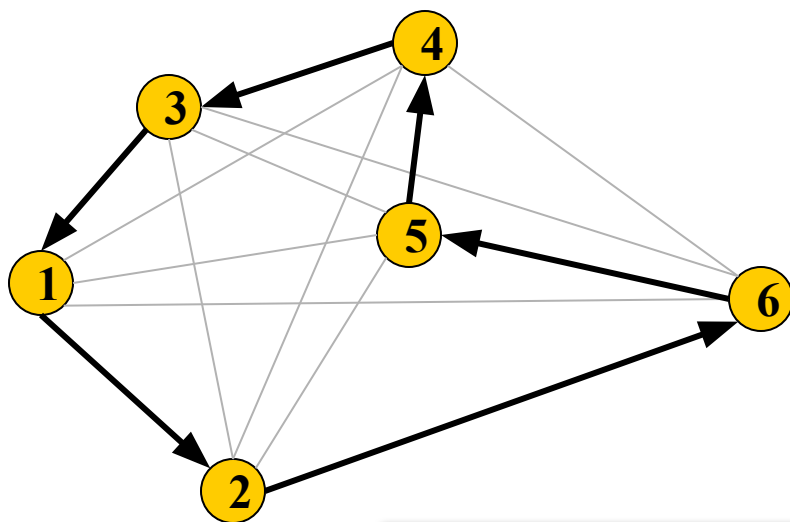
Симметричная задача

$$C_{ij} \neq C_{ji}$$

Несимметричная задача

Задача коммивояжёра. Точное решение

Алгоритм "перебора грубой силой" (brute-force enumeration)



C_{ij}	1	2	3	4	5	6
1		6	4	8	7	14
2	6		7	11	7	10
3	4	7		4	3	10
4	8	11	4		5	11
5	7	7	3	5		7
6	14	10	10	11	7	

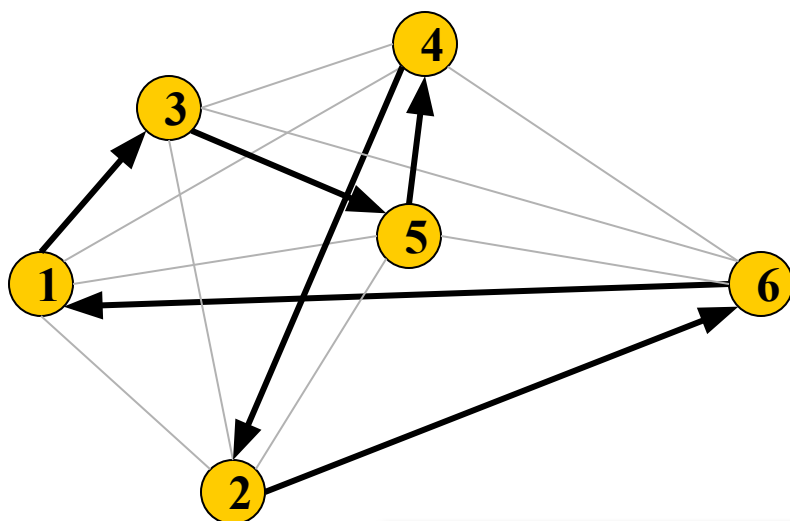
Решение 1-2-6-5-4-3-1, расстояние 36

Число вариантов в несимметричной задаче $(n-1)!$

5!	10!	15!	20!	25!	30!	35!	40!	45!	50!
$\sim 10^2$	$\sim 10^6$	$\sim 10^{12}$	$\sim 10^{18}$	$\sim 10^{25}$	$\sim 10^{32}$	$\sim 10^{40}$	$\sim 10^{47}$	$\sim 10^{56}$	$\sim 10^{64}$

Задача коммивояжёра. Быстрое решение

Алгоритм "ближайшего соседа"



C_{ij}	1	2	3	4	5	6
1		6	4	8	7	14
2	6		7	11	7	10
3	4	7		4	3	10
4	8	11	4		5	11
5	7	7	3	5		7
6	14	10	10	11	7	

Решение 1-3-5-4-2-6-1, расстояние 47

На каждом шаге алгоритма из всех возможных путей выбирается самый короткий. Как правило, на последних шагах приходится расплачиваться за "жадность" в начале. При определенном наборе данных алгоритм "ближайшего соседа" может даже выбрать наихудший путь.

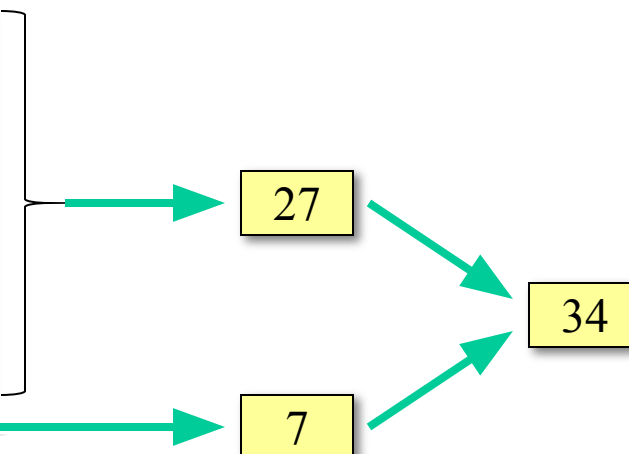
Задача коммивояжёра. Метод ветвей и границ

C_{ij}	1	2	3	4	5	6
1		6	4	8	7	14
2	6		7	11	7	10
3	4	7		4	3	10
4	8	11	4		5	11
5	7	7	3	5		7
6	14	10	10	11	7	

C_{ij}	1	2	3	4	5	6	
1		2	0	4	3	10	4
2	0		1	5	1	4	6
3	1	4		1	0	7	3
4	4	7	0		1	7	4
5	4	4	0	2		4	3
6	7	3	3	4	0		7

C_{ij}	1	2	3	4	5	6	
1		0	0	3	3	6	4
2	0		1	4	1	0	6
3	1	2		0	0	3	3
4	4	5	0		1	3	4
5	4	2	0	1		0	3
6	7	1	3	3	0		7
	0	2	0	1	0	4	

Приведение по строкам
Приведение по столбцам



Задача коммивояжёра. Метод ветвей и границ

Оценка нулей

C_{ij}	1	2	3	4	5	6
1		0	0	3	3	6
2	0		1	4	1	0
3	1	2		0	0	3
4	4	5	0		1	3
5	4	2	0	1		0
6	7	1	3	3	0	

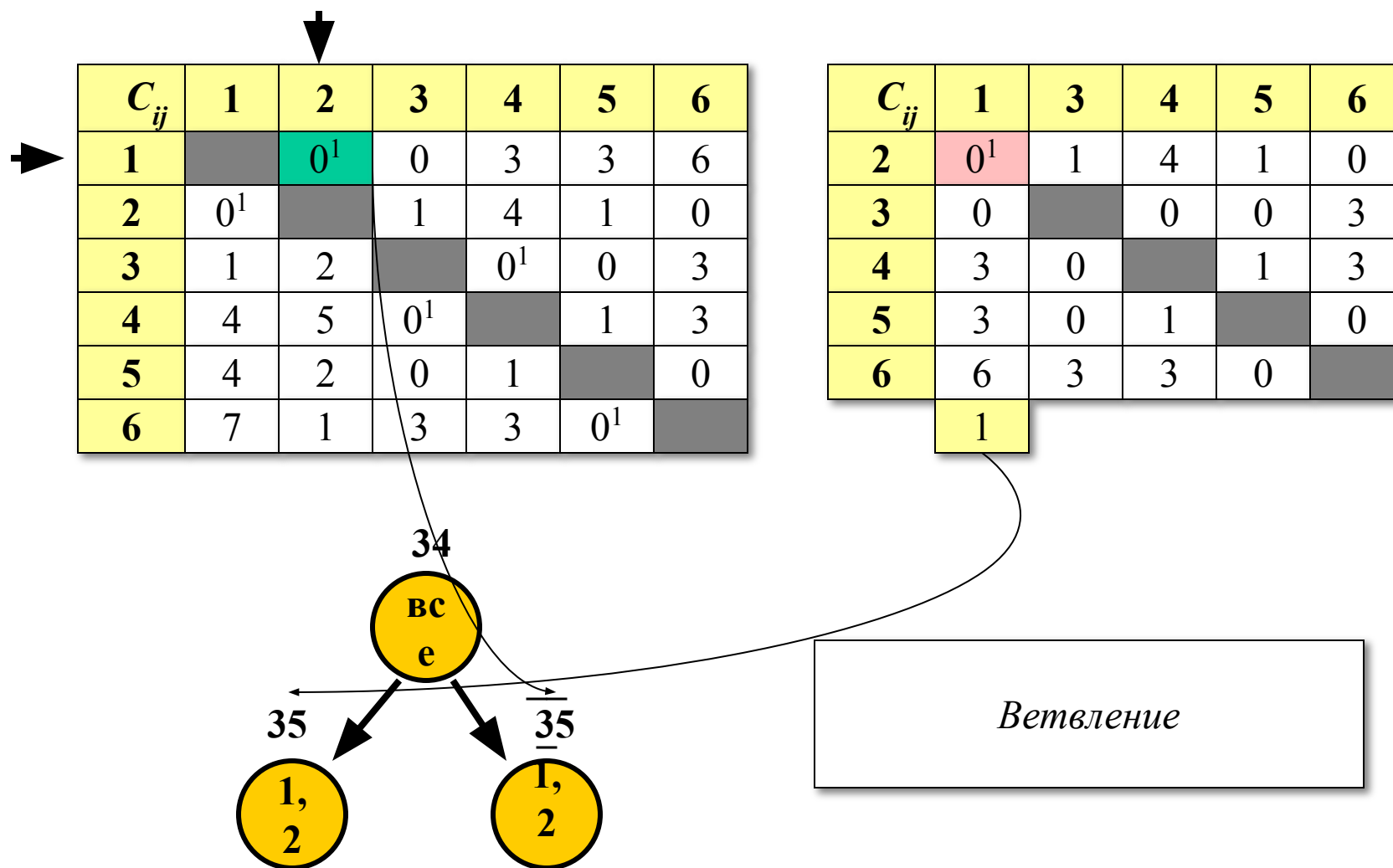
Отказ от пути 1-2 увеличит расстояние на 1 (1+0)

C_{ij}	1	2	3	4	5	6
1		0 ¹	0	3	3	6
2	0 ¹		1	4	1	0
3	1	2		0 ¹	0	3
4	4	5	0 ¹		1	3
5	4	2	0	1		0
6	7	1	3	3	0 ¹	

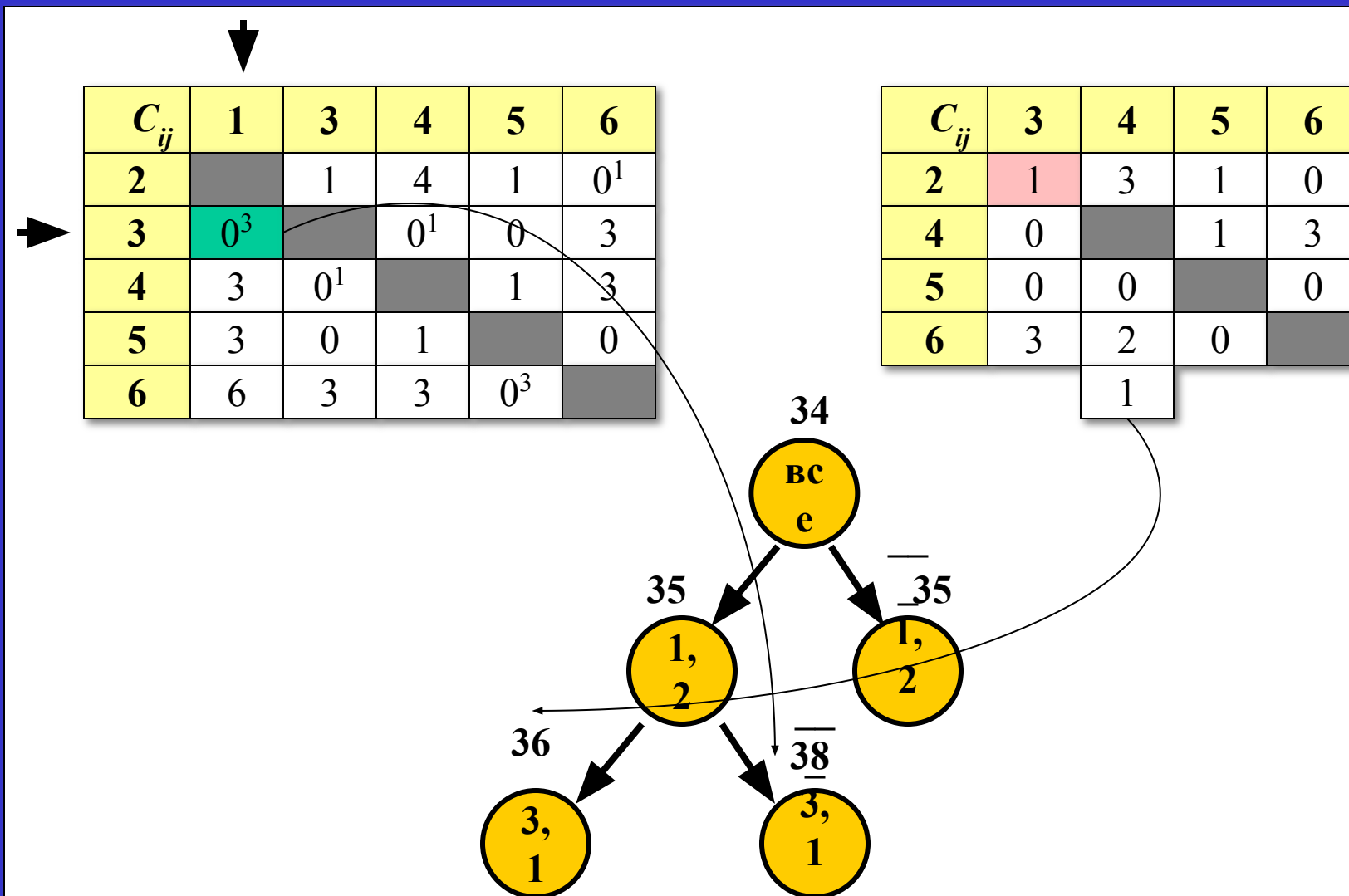
Выбирается первая из максимальных оценок

Все множество путей делится на два класса - включающие путь 1-2 и остальные

Задача коммивояжёра. Метод ветвей и границ



Задача коммивояжёра. Метод ветвей и границ

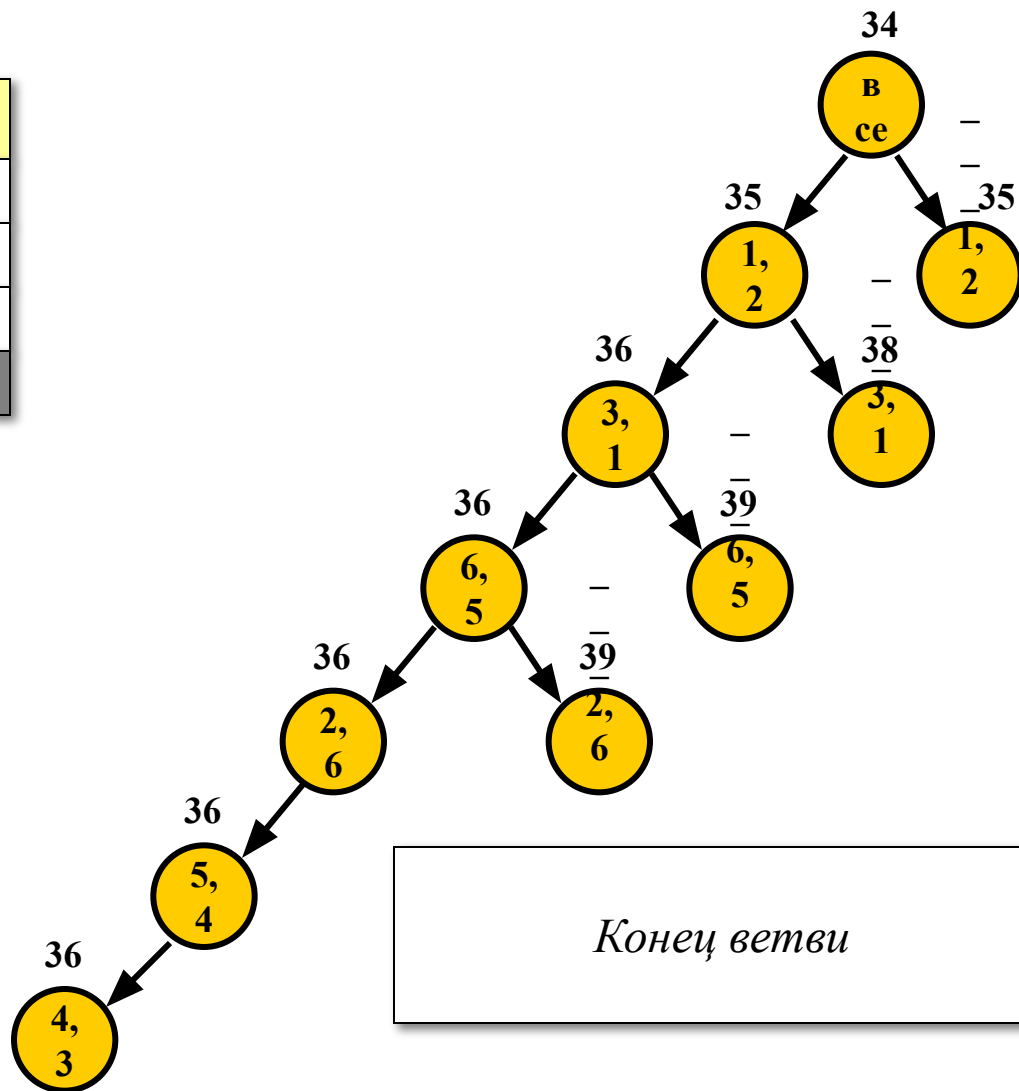


Задача коммивояжёра. Метод ветвей и границ

C_{ij}	3	4	5	6
2		3	1	0^1
4	0^1		1	3
5	0	0^2		0
6	3	2	0^3	

C_{ij}	3	4	6
2		3	0^6
4	0^3		3
5	0	0^3	

C_{ij}	3	4
4	0	
5		0

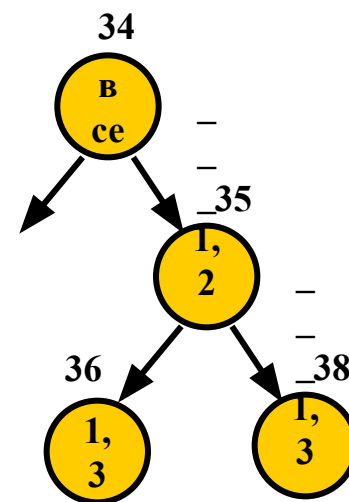


Задача коммивояжёра. Метод ветвей и границ

C_{ij}	1	2	3	4	5	6
1		0^1	0^3	3	3	6
2	0^1		1	4	1	0
3	1	2		0^1	0	3
4	4	5	0^1		1	3
5	4	2	0	1		0
6	7	1	3	3	0^1	

C_{ij}	1	2	4	5	6
2	0		4	1	0
3	1	1	0	0	3
4	4	4		1	3
5	4	1	1		0
6	7	0	3	0	

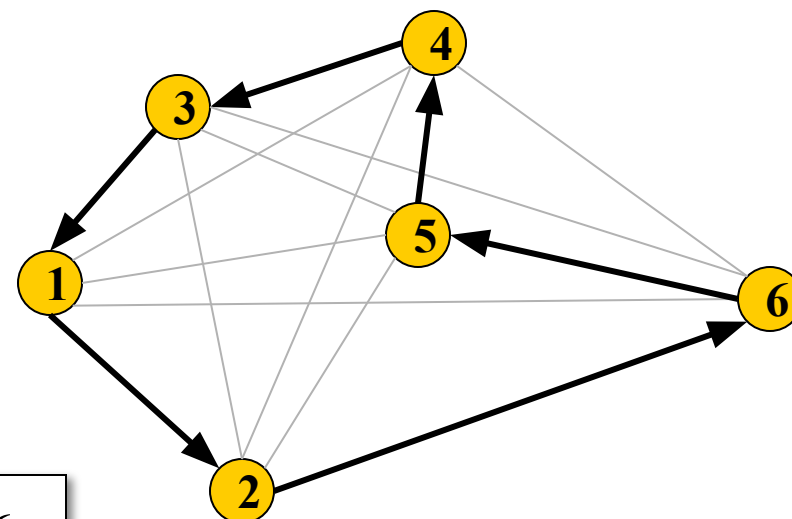
1



Проверка альтернативных
вариантов

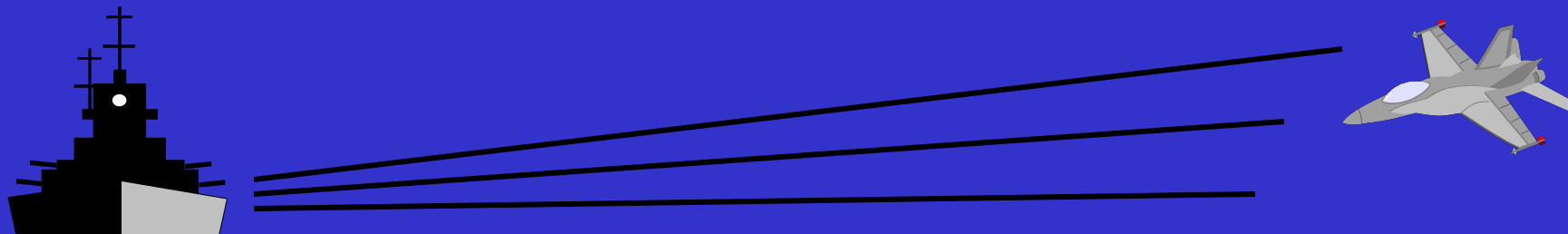
Задача коммивояжёра. Метод ветвей и границ

C_{ij}	1	2	3	4	5	6
1		6	4	8	7	14
2	6		7	11	7	10
3	4	7		4	3	10
4	8	11	4		5	11
5	7	7	3	5		7
6	14	10	10	11	7	



Решение 1-2-6-5-4-3-1, расстояние 36

Динамическое программирование



Метод динамического программирования состоит в том, что оптимальное управление строится постепенно. На каждом шаге оптимизируется управление только этого шага. Вместе с тем на каждом шаге управление выбирается с учетом последствий, так как управление, оптимизирующее целевую функцию только для данного шага, может привести к неоптимальному эффекту всего процесса. Управление на каждом шаге должно быть оптимальным с точки зрения процесса в целом.

Каково бы ни было начальное состояние системы перед очередным шагом, управление на этом этапе выбирается так, чтобы выигрыш на данном шаге плюс оптимальный выигрыш на всех последующих шагах был максимальным.

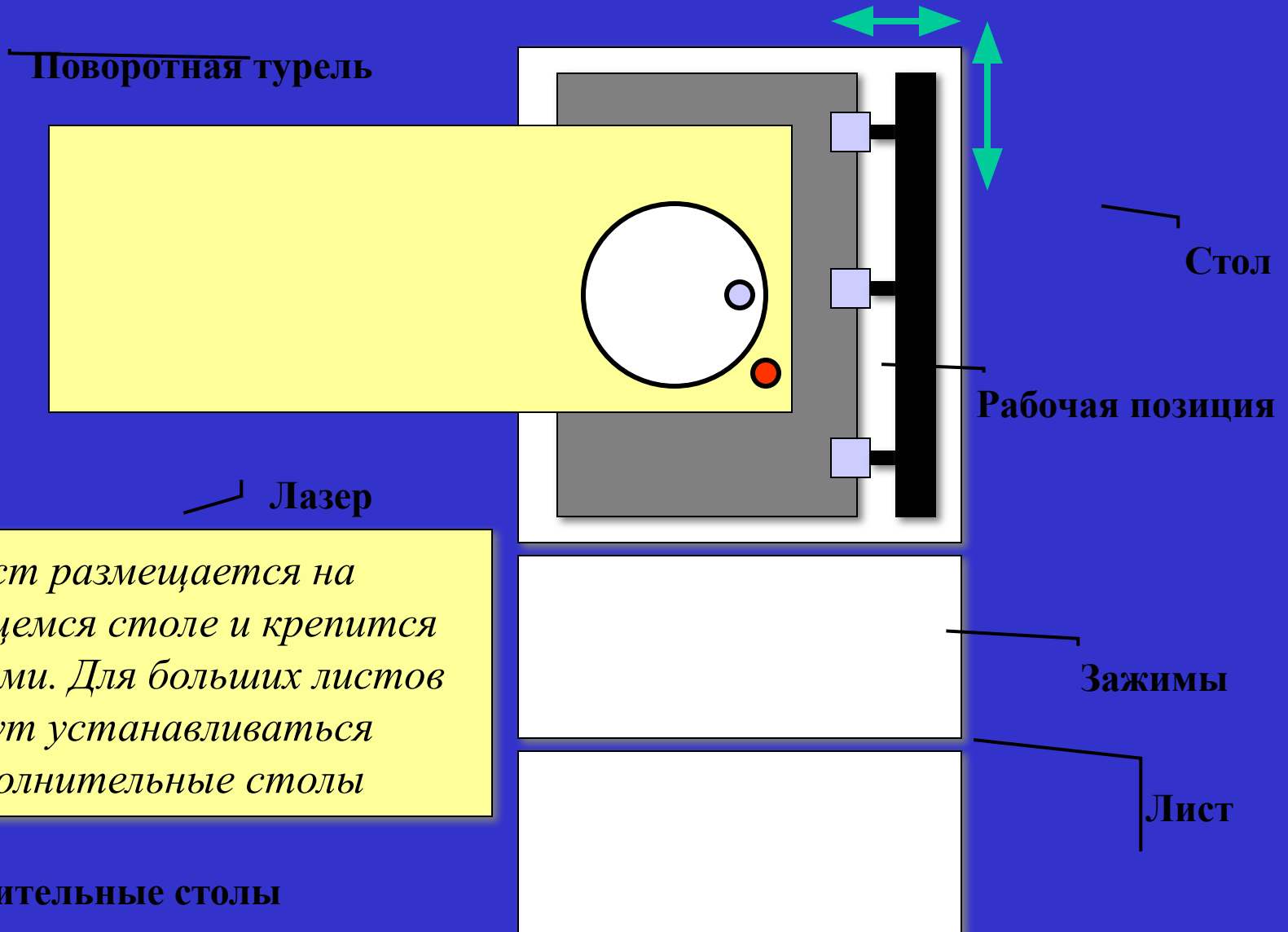
Ричард Эрнст Беллман (1920 - 1984)

Пример задачи динамического программирования

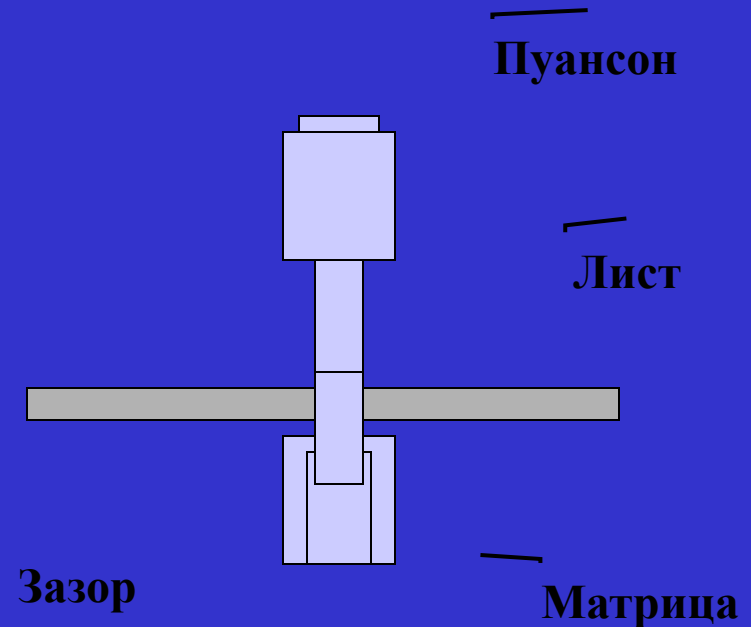


*Оптимизация последовательности работ
штамповочной машины Strippit-1250*

Схема работы машины

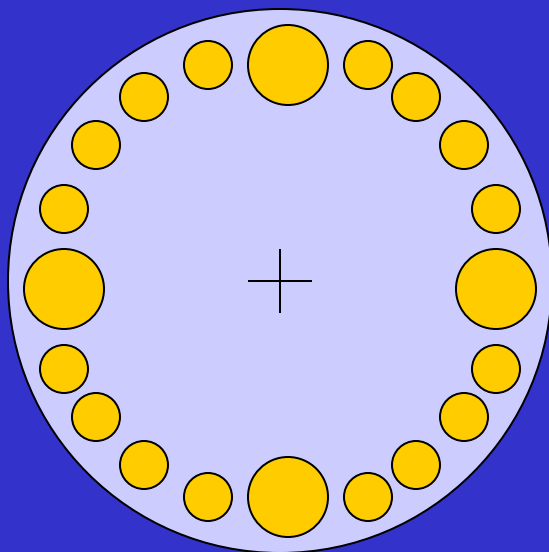


Пробивной инструмент



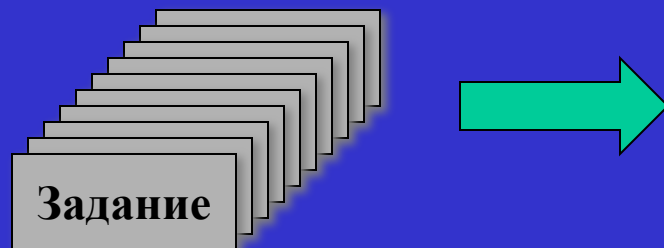
Для пробивки отверстий используется пара инструментов - пуансон и матрица. Для одного пуансона могут применяться разные матрицы. Выбор матрицы зависит от толщины листа. Чем толще лист, тем больше должен быть зазор между пуансоном и матрицей.

Поворотная турель



Пробивной инструмент (пуансоны и матрицы) устанавливается в поворотную турель, имеющую 20 станций (16 нормальных и 4 большие)

Задача оптимизации последовательности заданий



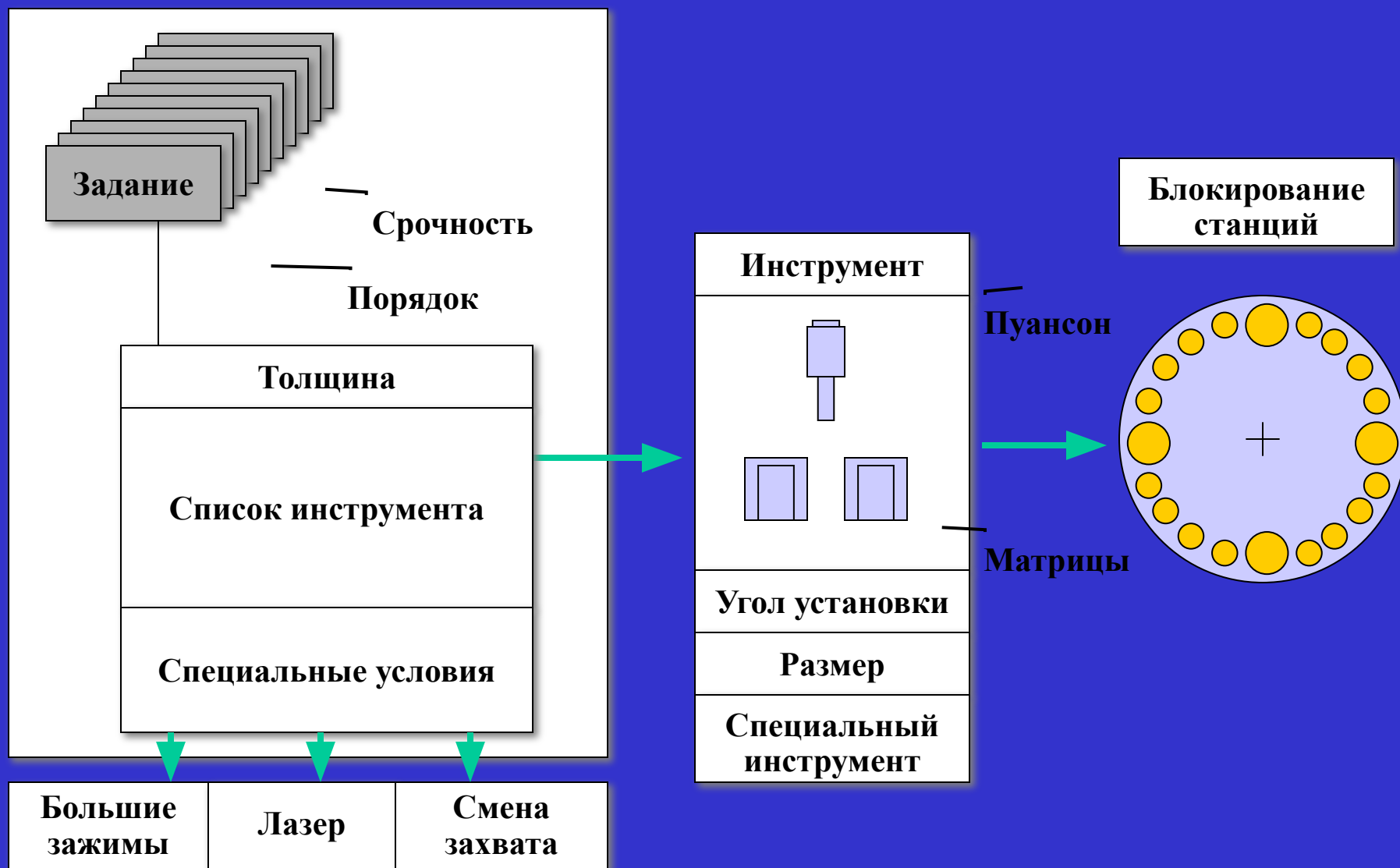
В малосерийном производстве через штамповочную машину за день проходит до 50 партий изделий (заданий). Для каждого задания требуется установка в турель от 1 до 19 инструментов. Для минимизации общего времени производства требуется подобрать такую последовательность выполнения заданий, при которой время смены инструмента было бы минимальным.

Общее время производства

**Выполнение
программы**

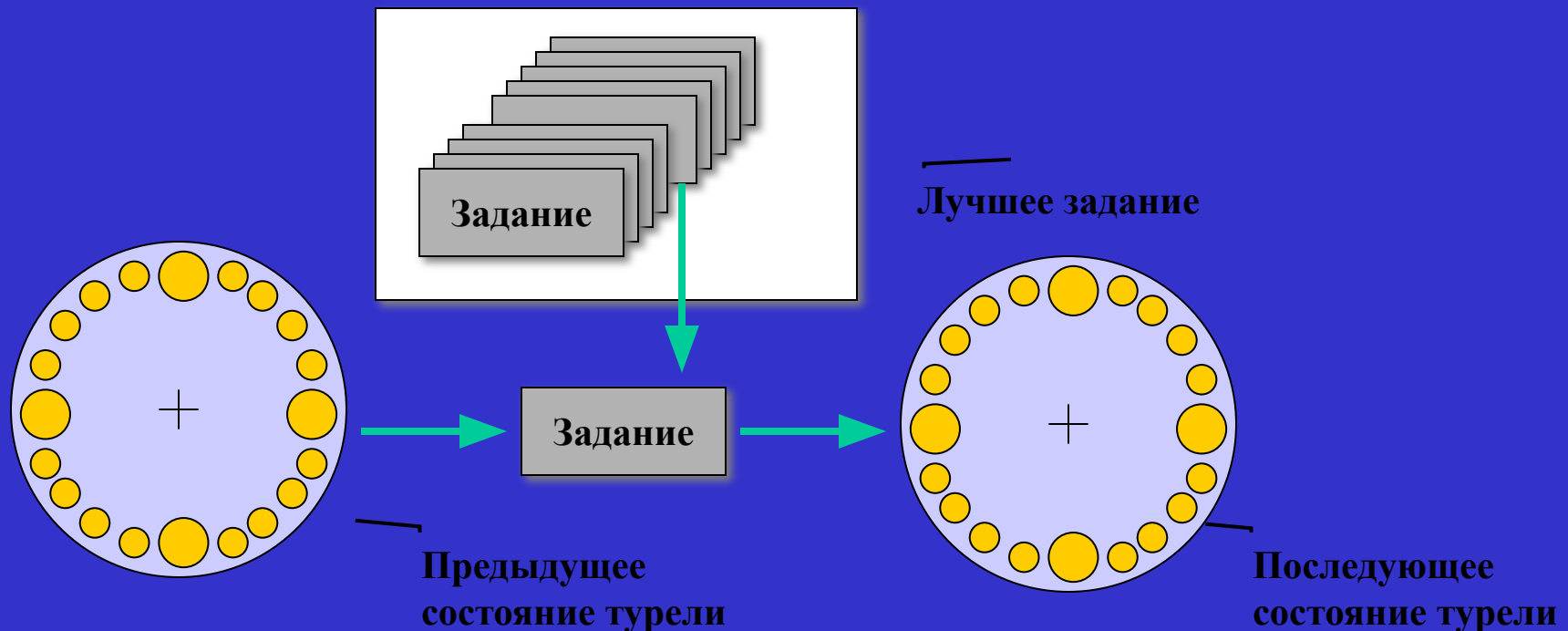
**Смена
инструмента**

Ограничения



Эвристический алгоритм

1. Последовательно выполняются N шагов, где N - число заданий.
2. На каждом шаге из списка оставшихся заданий выбирается лучшее задание.
3. Выбор на основе стоимостной функции для всех комбинаций инструмент - станция.
4. При вычислении стоимостной функции учитывается влияние выбора на оставшиеся задания.
5. Ограничения применяются как можно раньше для снижения размерности задачи.



Учет ограничений

Задания



1	2	3	4	5	6	7	8
		●		●			
		●		●			
●			●			●	●
●			●		●	●	●
●	●		●		●	●	●
●	●		●		●		●
●	●		●				●

3 и 5 - срочные задания

6 перед 2

7 перед 6

Станции

Большой инструмент

●					●					●					●				
---	--	--	--	--	---	--	--	--	--	---	--	--	--	--	---	--	--	--	--

Малый инструмент

	●	●	●	●		●	●	●	●		●	●	●	●		●	●	●	●
--	---	---	---	---	--	---	---	---	---	--	---	---	---	---	--	---	---	---	---

Угол 0°

●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Угол 45°

																		●	
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	---	--

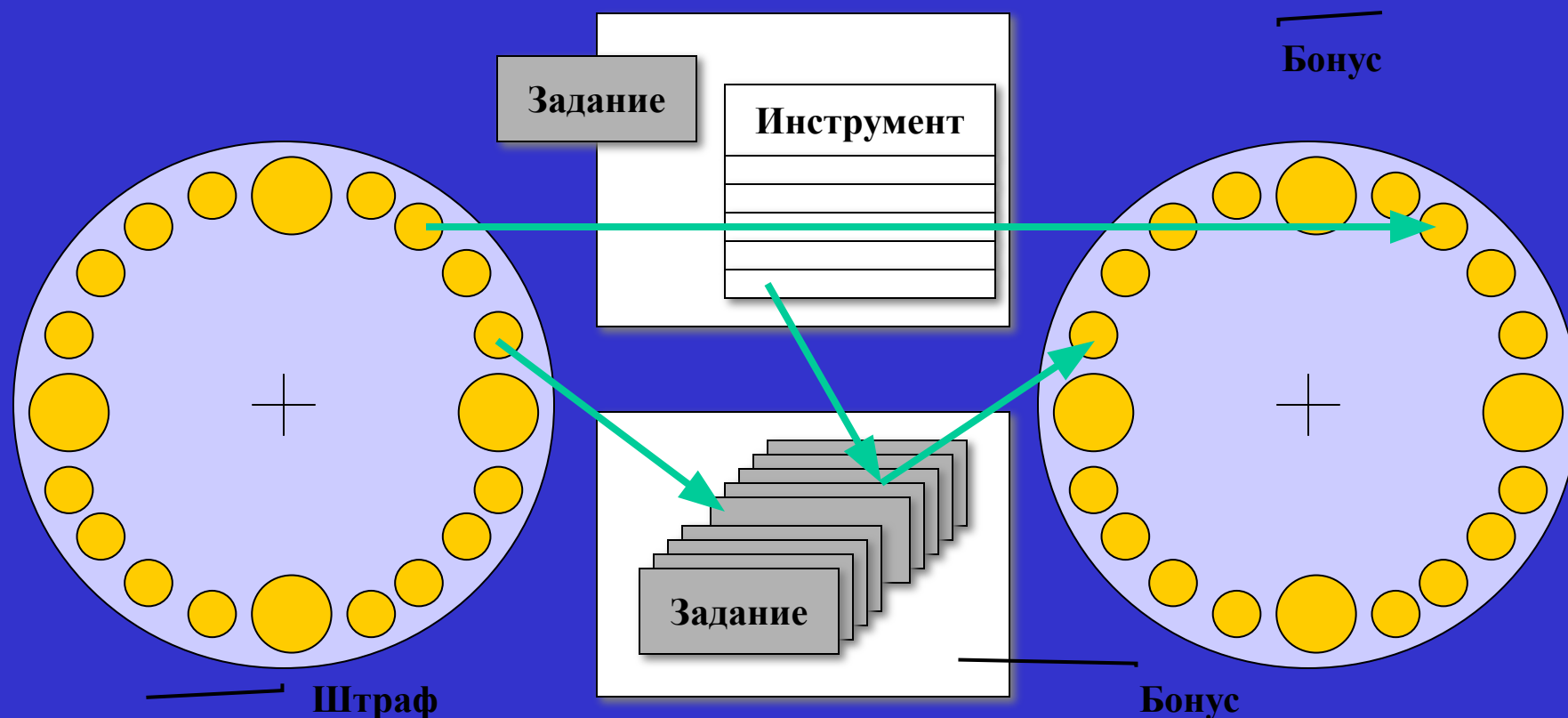
Угол 90°

●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--	---

Специальный инструмент

																		●	
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	---	--

Расчет стоимостной функции смены инструмента



Стоимостная функция поощряет установку инструмента, который может быть использован другими заданиями, и наказывает снятие инструмента, для которого еще есть задания.

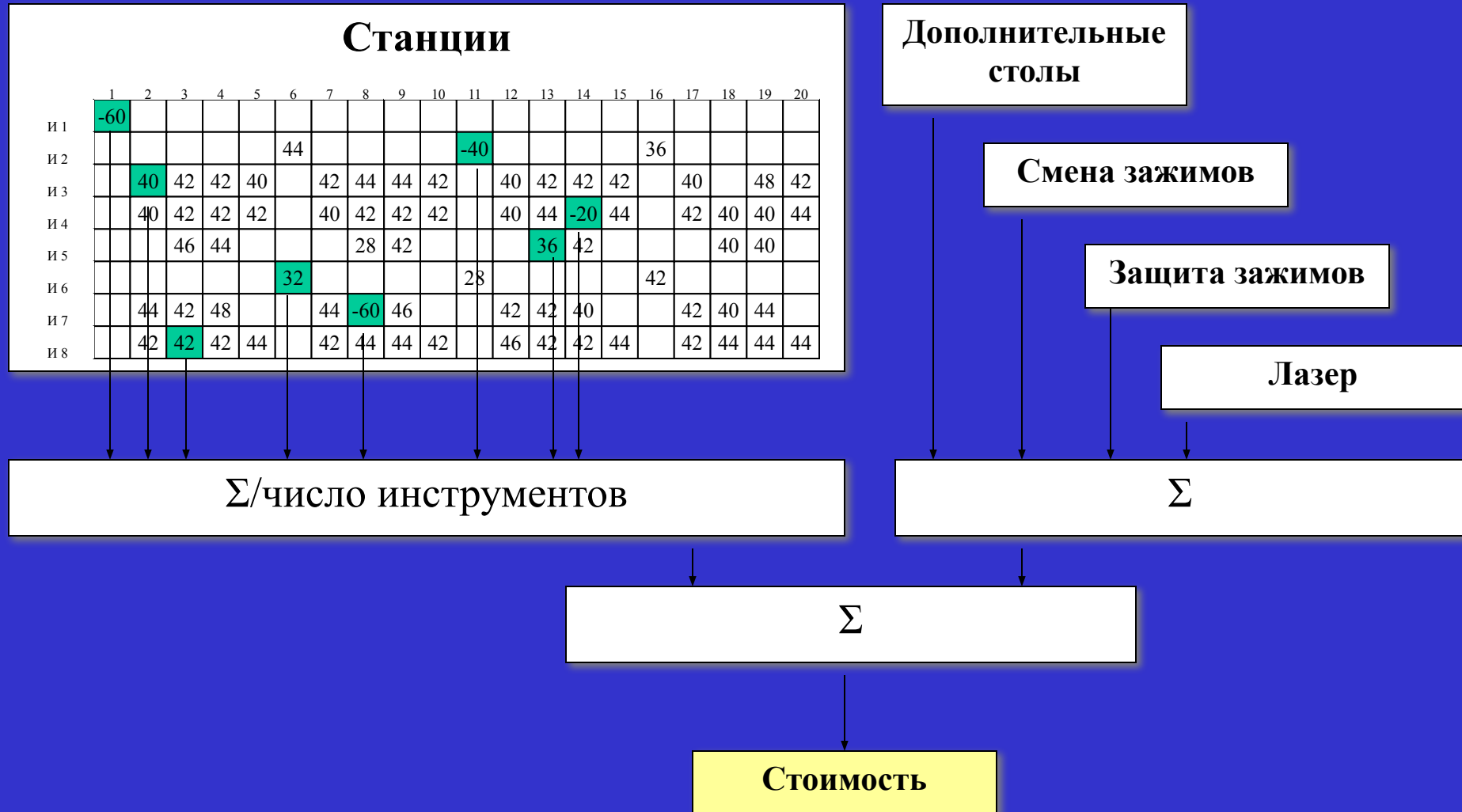
Пример стоимостной функции для 8 инструментов

Станции

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
И 1	-60																			
И 2						44					-40					36				
И 3		40	42	42	40		42	44	44	42		40	42	42	42		40		48	42
И 4		40	42	42	42		40	42	42	42		40	44	-20	44		42	40	40	44
И 5			46	44				28	42				36	42				40	40	
И 6						32					28					42				
И 7		44	42	48			44	-60	46			42	42	40			42	40	44	
И 8		42	42	42	44		42	44	44	42		46	42	42	44		42	44	44	44

Ищется вариант размещения инструментов, который даст минимальное значение стоимостной функции

Полная стоимостная функция



Выбор решения

Шаг 3

1	2	3	4	5	6	7	8
				●			
		●					
●			●			●	●
●			●		●	●	●
●	●		●		●	●	●
●	●		●		●	●	●
●	●		●				●

Минимальная
стоимость

Шаг 4

1	2	3	4	5	6	7	8
				●			
		●					
							●
●			●			●	●
●			●		●	●	●
●	●		●		●	●	●
●	●		●		●		●
●	●		●				●

Результаты оптимизации

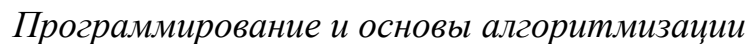
```
INIT STATE-----
Punch [ 0 287 292 281 35 302 316 302 44 81 17 316 45 249 14 133 169 323 281 275 ] 0: 0
Matrix [ 0 459 476 442 28 507 535 495 37 78 15 535 43 364 8 134 206 550 442 423 ] 0: 0
Angle [ 0 90 0 0 90 0 0 0 0 0 0 90 0 90 0 0 0 0 45 0 0 ]
```

```
RETOOLING-----
Punch [ 287 91 40 281 138 316 323 302 44 289 17 316 45 249 14 133 169 281 281 275 ] 2: 6
Matrix [ 458 85 35 442 142 531 548 495 37 463 15 535 43 364 8 134 206 440 442 423 ] 2: 6
Angle [ 0 0 0 0 0 90 0 0 0 90 90 0 90 0 0 0 0 45 0 0 ]
```

```
Job 15      *      *      *
Job 14      *      *      *
Job 16      *      *      *
Job 17      *      *      *
Job 18      *      *      *
Job 19      *      *      *
Job 20      *      *      *
Job 4              *      *      *      *
Job 3  *              *      *      *
```

```
RETOOLING-----
Punch [ 287 91 40 281 138 316 323 302 44 60 316 139 6 17 14 133 54 297 302 275 ] 1: 7
Matrix [ 458 85 35 440 142 531 548 495 37 53 531 143 2 10 8 134 47 484 495 423 ] 1: 8
Angle [ 0 0 0 0 0 90 0 0 0 0 0 0 0 0 0 0 0 0 90 0 ]
```

```
Job 1              *      *      *      *      *      *
Job 2              *      *      *      *      *
```



Результаты оптимизации

RETOOLING-----																							
Punch	[	287	91	40	281	138	316	323	302	44	60	316	139	6	110	14	133	54	297	302	275	]	0: 1
Matrix	[	458	85	35	440	142	535	550	495	37	53	535	143	2	111	8	134	47	484	495	423	]	2: 2
Angle	[	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	90	0	]	

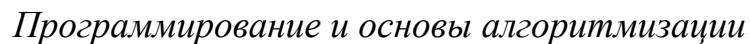
Job 13 * * *

RETOOLING-----																							
Punch	[	287	91	40	281	138	316	323	302	44	60	316	139	1	110	14	169	155	35	302	275	]	1: 3
Matrix	[	458	85	33	440	142	531	550	495	37	53	531	143	2	111	8	206	172	28	495	423	]	3: 3
Angle	[	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	90	0	]	

Job	5								*
Job	6								*
Job	7								*
Job	21	*		*		*		*	
Job	28				*		*	*	* *
Job	27		*	*		*		*	* *

RETOOLING																							
Punch	[	287	91	40	281	249	163	323	302	70	60	316	139	1	25	14	169	155	35	302	133	]	1: 4
Matrix	[	459	85	33	440	364	191	549	495	66	53	531	143	1	21	8	206	173	28	495	134	]	2: 7
Angle	[	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	]	

Job	8									*
Job	9									*
Job	10						*			*
Job	11			*						
Job	12			*						
Job	25						*			
Job	24		*		*			*		*
Job	23	*			*	*			*	*



Результаты оптимизации

RETOOLING-----																								
Punch	[	287	91	40	281	249	163	323	302	70	60	316	139	1	25	14	169	155	35	302	133	]	0 :	0
Matrix	[	459	85	33	440	364	191	548	495	66	53	531	143	1	21	8	206	173	28	495	134	]	0 :	1
Angle	[	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	]		

Job 22 * *

[illegible]

Job 33 * |

Job 30 * * | * * *

[illegible]

Job 31		*	*	*	*	*	*		*		*	*		*	*
Job 34	*				*		*		*				*	*	*
Job 32			*			*	*	*	*	*		*			

RETOOLING																							
Punch	[287 120	40	36	105	311	323	127	91	144	284	139	1	45	97	311	51	157	321	112	]	2 :	6	
Matrix	[459 120	39	36	106	523	551	127	88	153	453	145	1	45	98	523	50	177	546	114	]	2 :	8	
Angle	[0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	]			

[illegible]