e-Companion

Exact Solution of the Single Picker Routing Problem with Scattered Storage

Appendix A: Dynamic-Programming Transitions for Single-Block Parallel-Aisle Warehouse Layout

Tables EC.1 and EC.2 list all transitions between the stages of the dynamic program for the single-block parallel-aisle warehouse layout. Contrary to the state space for the classical SPRP by Ratliff and Rosenthal (1983), for the SPRP-SS, the state space consists of several transitions `top`(i), `bottom`(i), and `gap`(h,i) if two or more article can be picked in one aisle.

Table EC.1 Possible transitions from j^- to j^+.

	Transition via					
State	`1pass`	`top`(i)	`bottom`(i)	`gap`(h,i)	`2pass`	`void`
UU1c	EE1c	UU1c	UU1c	UU1c	UU1c	UU1c
E01c	UU1c	E01c	EE2c	EE2c	EE1c	E01c
0E1c	UU1c	EE2c	0E1c	EE2c	EE1c	0E1c
EE1c	UU1c	EE1c	EE1c	EE1c	EE1c	EE1c
EE2c	UU1c	EE2c	EE2c	EE2c	EE1c	EE2c
000c	UU1c	E01c	0E1c	EE2c	EE1c	000c
001c	–	–	–	–	–	001c

Table EC.2 Possible transitions from j^+ to j^-.

	Transition via				
State	`11`	`20`	`02`	`22`	`00`
UU1c	UU1c	–	–	–	–
E01c	–	E01c	–	EE2c	001c
0E1c	–	–	0E1c	EE2c	001c
EE1c	–	E01c	0E1c	EE1c	001c
EE2c	–	–	–	EE2c	–
000c	–	E01c	0E1c	–	000c
001c	–	–	–	–	001c

Appendix B: Instance Generation Procedure for SPRP-SS Instances

The generation procedure described by Weidinger and Goeke and Schneider completely fills the warehouse with articles. The assumption is that one article is stored in each of the $m \cdot C$ possible pick positions. In particular, opposite cells in an aisle are considered as one pick position.

First, the scatter factor α can be realized by defining a set of $\xi = \lceil mC/\alpha \rceil \geq n$ different articles stored in the warehouse (not all of them are in the final pick list). To ensure that exactly ξ different articles are available, each of them is randomly assigned to a unique pick position.

Second, all ξ different articles are divided into the three classes A, B, and C (in practice depending on the turnover rate) with 20% in class A, 30% in class B, and 50% in class C. The ABC class-based addition of further SKUs must now ensure that (on average) 80% of the positions are stocked with A-class articles, 15% with B-class articles, and 5% with C-class articles. One by one, articles are assigned to positions as follows: According to the 80-15-5 distribution, a class is chosen. Then, a random article of that class is selected. Finally, this article is assigned randomly to one of the (up to this point) free storage positions in the warehouse.

Third, supply values are then determined with a further distinction between instances with

- a *sufficient number of items* or *unit demand*: Here, the demand and supply values are set to $q_s = b_{sp} = 1$ for all $s \in S$ and $p \in P_s$;
- *general supply and demand*: Here, the supply available at a position is randomly drawn as $b_{sp} \in \{1, 2, 3\}$. Moreover, demands are randomly drawn as $q_s \in \{1, \ldots, \min(6, \sum_p b_{sp})\}$.

Finally, the pick list is generated so that it contains exactly n different articles. Iteratively, a pick position is randomly drawn. If the SKU at this position is not yet an article in the pick list, this article s is chosen to be included in the pick list.

Appendix C: Results for SPRP-SS Instances with General Supply and Demand

For the sake of brevity, Tables EC.3 and EC.4 have been moved to the e-Companion. The former is structurally identical to Table 5 but for general supply and demand. The latter reports average computation times for the NF model, again for general supply and demand.

Table EC.3 Comparison of NF and GS with computation times (in milliseconds) and speedup factor comparing GS and NF for the SPRP-SS, a single-block parallel-aisle warehouse, and general supply and demand.

Scatter factor	Warehouse dimension	Number of SKUs in pick list $n=3$		$n=7$		$n=15$		$n=30$	
	(m,C)	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$
$\alpha=2$	$(5,30)$	46.8	4.5	67.0	2.0	87.9	1.1	107.9	0.9
	$(5,60)$	47.4	5.1	56.1	2.7	99.0	1.2	127.2	0.9
	$(5,180)$	59.4	6.6	71.8	2.3	106.7	1.0	167.1	0.8
	$(10,30)$	70.8	5.4	92.7	2.3	189.7	1.4	231.5	1.0
	$(10,60)$	56.7	9.6	78.9	2.9	159.4	1.6	309.2	0.9
	$(10,180)$	71.4	6.9	87.8	2.7	147.4	1.3	344.5	0.8
	$(25,30)$	78.6	5.3	122.6	3.4	296.5	2.0	712.3	1.4
	$(25,60)$	87.1	8.6	134.1	3.9	268.7	2.1	589.2	1.3
	$(25,180)$	111.7	8.8	113.4	6.2	300.0	2.1	643.4	1.5
	$(50,30)$	162.9	4.6	226.6	5.8	415.8	3.1	847.3	1.9
	$(50,60)$	146.6	7.2	228.4	7.1	387.1	3.3	895.1	2.2
	$(50,180)$	199.1	16.1	214.5	12.5	416.0	4.5	931.3	2.5
$\alpha=5$	$(10,30)$	226.4	3.4	505.8	1.5	794.4	0.8	1021.3	0.6
	$(10,60)$	173.6	4.8	448.6	2.4	859.4	1.0	2183.4	0.5
	$(10,180)$	134.0	5.5	396.8	3.0	1070.6	1.1	2980.6	0.5
	$(25,30)$	558.2	4.5	1190.9	2.8	1913.9	1.1	4141.6	0.6
	$(25,60)$	552.2	4.7	1229.1	2.7	2210.2	1.4	4174.6	0.6
	$(25,180)$	439.6	7.7	912.3	4.5	1372.5	1.8	4862.5	0.7
	$(50,30)$	667.3	4.2	1186.1	2.9	1369.5	2.8	4398.7	1.3
	$(50,60)$	887.7	6.3	1021.8	5.1	1865.1	3.4	5889.4	1.3
	$(50,180)$	461.9	8.6	964.9	9.3	1755.8	4.4	5620.5	1.6
$\alpha=10$	$(25,30)$	2022.4	4.0	3635.7	1.5	6265.0	0.7	8669.6	0.6
	$(25,60)$	1657.2	3.2	6333.0	1.7	9376.2	0.9	10 845.0	0.6
	$(25,180)$	914.0	5.2	3534.5	2.5	9228.1	1.5	13 239.4	0.8
	$(50,30)$	3643.5	1.4	3959.6	2.3	10 193.5	1.1	17 616.0	0.8
	$(50,60)$	2307.5	3.5	3425.4	3.3	9315.7	1.3	17 902.1	0.9
	$(50,180)$	1599.1	4.8	4260.0	6.1	11 015.0	2.7	20 612.2	1.7

Table EC.4 Computation times (in milliseconds) of NF for the SPRP-SS, two-block parallel-aisle warehouse, and general supply and demand.

Scatter factor	Warehouse dimension	Number of articles in pick list $n=3$	$n=7$	$n=15$	$n=30$
	(m,C)	t_{NF}	t_{NF}	t_{NF}	t_{NF}
$\alpha=2$	$(5,30)$	21.6	44.9	78.3	119.1
	$(5,60)$	24.7	46.4	103.7	177.6
	$(5,180)$	34.3	64.8	105.1	258.9
	$(10,30)$	52.3	81.0	173.4	276.3
	$(10,60)$	55.4	86.8	163.8	408.9
	$(10,180)$	75.6	85.0	128.4	385.8
	$(25,30)$	82.1	135.5	278.5	678.4
	$(25,60)$	63.9	119.9	285.5	680.0
	$(25,180)$	72.5	144.7	322.8	722.1
	$(50,30)$	156.3	579.3	453.4	945.6
	$(50,60)$	110.7	299.2	419.2	1269.0
	$(50,180)$	211.6	215.2	480.0	2295.1
$\alpha=5$	$(10,30)$	537.5	1147.4	1453.7	2237.4
	$(10,60)$	836.2	1448.4	1882.0	4056.2
	$(10,180)$	891.9	1383.6	3353.7	6011.4
	$(25,30)$	1697.3	3076.7	4161.3	6685.3
	$(25,60)$	1871.9	4331.6	4074.5	8855.6
	$(25,180)$	2336.7	2144.2	4066.7	12 807.7
	$(50,30)$	8172.5	6566.0	9222.6	24 512.2
	$(50,60)$	3376.1	4007.4	12 136.5	60 624.2
	$(50,180)$	3350.2	7916.7	28 560.3	77 918.1
$\alpha=10$	$(25,30)$	8774.1	9433.0	12 728.5	25 128.6
	$(25,60)$	12 120.0	28 881.6	28 692.2	32 253.0
	$(25,180)$	11 025.8	25 136.3	33 230.2	32 516.8
	$(50,30)$	15 735.8	56 414.1	179 451.1	110 293.4
	$(50,60)$	25 184.1	57 259.9	78 620.4	437 665.8
	$(50,180)$	18 090.0	39 523.2	147 424.7	350 290.7

Appendix D: Dynamic-Programming State Space for Two-Block Parallel-Aisle Warehouse Layout

The dynamic program of Roodbergen and de Koster (2001b) solving the SPRP for two-block parallel-aisle warehouse has $3m+1$ stages. For each aisle $j \in J = \{1, 2, \ldots, m\}$, the stage j^- describes a PTS before aisle j is traversed, the stage j^{+x} when the back ('x') block in aisle j is traversed, and the stage j^{+y} when the front ('y') block in aisle j is traversed. An artificial stage $(m+1)^-$ is added as in the single-block case.

The states of the DP describe vertex degrees and the connectivity of the PTS. Since there are three crossing points per aisle now (in the back cross-aisle, middle cross-aisle, front cross-aisle), the vertices a_j, b_j, and c_j are introduced for this purpose. As a result, there are 25 states relevant to construct an optimal picker tour:

$$\mathscr{S} = \{\texttt{0000c}, \texttt{0001c}, \texttt{E001c}, \texttt{0E01c}, \texttt{00E1c}, \texttt{EE01c}, \texttt{E0E1c}, \texttt{0EE1c}, \texttt{EEE1c}, \texttt{UU01c}, \texttt{U0U1c}, \texttt{0UU1c}, \texttt{EUU1c},$$
$$\texttt{UEU1c}, \texttt{UUE1c}, \texttt{EE02c}, \texttt{E0E2c}, \texttt{0EE2c}, \texttt{EEE2c.a-bc}, \texttt{EEE2c.b-ac}, \texttt{EEE2c.c-ab}, \texttt{EUU2c}, \texttt{UEU2c}, \texttt{UUE2c}, \texttt{EEE3c}\}$$

(the first three symbols again describe vertex degrees for the back/middle/front cross aisle). The connectivity information is more detailed compared to the DP of Ratliff and Rosenthal: there can be one (`1c`), two (`2c`), or three (`3c`) connected components. Moreover, if all three vertices a_j, b_j, and c_j are even (EEE), two connected components can result from having middle and front cross-aisle connected (`a-bc`), back and front cross-aisle connected (`b-ac`), or back and middle cross-aisle connected (`c-ab`). Transitions between stages are summarized in Tables EC.5, EC.6, and EC.7. Note that the state space consists of several transitions `top`(i), `bottom`(i), and `gap`(h,i) if two or more article can be picked in one aisle.

Table EC.5 Possible transitions from j^- to j^{+y}.

	Transition via					
State	`1pass`	`top`(i)	`bottom`(i)	`gap`(h,i)	`2pass`	`void`
0000c	UU01c	E001c	0E01c	EE02c	EE01c	0000c
0001c	–	–	–	–	–	0001c
E001c	UU01c	E001c	EE02c	EE02c	EE01c	E001c
0E01c	UU01c	EE02c	0E01c	EE02c	EE01c	0E01c
00E1c	UUE2c	E0E2c	0EE2c	EEE3c	EEE2c.c-ab	00E1c
EE01c	UU01c	EE01c	EE01c	EE01c	–	EE01c
E0E1c	UUE1c	E0E1c	EEE2c.b-ac	EEE2c.b-ac	EEE1c	E0E1c
0EE1c	UUE1c	EEE2c.a-bc	0EE1c	EEE2c.a-bc	EEE1c	0EE1c
EEE1c	UUE1c	EEE1c	EEE1c	EEE1c	–	EEE1c
UU01c	EE01c	UU01c	UU01c	UU01c	–	UU01c
U0U1c	EUU1c	U0U1c	UEU2c	UEU2c	UEU1c	U0U1c
0UU1c	UEU1c	EUU2c	0UU1c	EUU2c	EUU1c	0UU1c
EUU1c	UEU1c	EUU1c	EUU1c	EUU1c	–	EUU1c
UEU1c	EUU1c	UEU1c	UEU1c	UEU1c	–	UEU1c
UUE1c	EEE1c	UUE1c	UUE1c	UUE1c	–	UUE1c
EE02c	UU01c	EE02c	EE02c	EE02c	EE01c	EE02c
E0E2c	UUE2c	E0E2c	EEE3c	EEE3c	EEE2c.c-ab	E0E2c
0EE2c	UUE2c	EEE3c	0EE2c	EEE3c	EEE2c.c-ab	0EE2c
EEE2c.a-bc	UUE1c	EEE2c.a-bc	EEE2c.a-bc	EEE2c.a-bc	EEE1c	EEE2c.a-bc
EEE2c.b-ac	UUE1c	EEE2c.b-ac	EEE2c.b-ac	EEE2c.b-ac	EEE1c	EEE2c.b-ac
EEE2c.c-ab	UUE2c	EEE2c.c-ab	EEE2c.c-ab	EEE2c.c-ab	–	EEE2c.c-ab
EUU2c	UEU1c	EUU2c	EUU2c	EUU2c	EUU1c	EUU2c
UEU2c	EUU1c	UEU2c	UEU2c	UEU2c	UEU1c	UEU2c
UUE2c	EEE2c.c-ab	UUE2c	UUE2c	UUE2c	–	UUE2c
EEE3c	UUE2c	EEE3c	EEE3c	EEE3c	EEE2c.c-ab	EEE3c

Table EC.6 Possible transitions from j^{+y} to j^{+x}.

	Transition via					
State	`1pass`	`top`(i)	`bottom`(i)	`gap`(h,i)	`2pass`	`void`
0000c	0UU1c	0E01c	00E1c	0EE2c	0EE1c	0000c
0001c	–	–	–	–	–	0001c
E001c	EUU2c	EE02c	E0E2c	EEE3c	EEE2c.a-bc	E001c
0E01c	0UU1c	0E01c	0EE2c	0EE2c	0EE1c	0E01c
00E1c	0UU1c	0EE2c	00E1c	0EE2c	0EE1c	00E1c
EE01c	EUU1c	EE01c	EEE2c.c-ab	EEE2c.c-ab	EEE1c	EE01c
E0E1c	EUU1c	EEE2c.b-ac	E0E1c	EEE2c.b-ac	EEE1c	E0E1c
0EE1c	0UU1c	0EE1c	0EE1c	0EE1c	–	0EE1c
EEE1c	EUU1c	EEE1c	EEE1c	EEE1c	–	EEE1c
UU01c	UEU1c	UU01c	UUE2c	UUE2c	UUE1c	UU01c
U0U1c	UUE1c	UEU2c	U0U1c	UEU2c	UEU1c	U0U1c
0UU1c	0EE1c	0UU1c	0UU1c	0UU1c	–	0UU1c
EUU1c	EEE1c	EUU1c	EUU1c	EUU1c	–	EUU1c
UEU1c	UUE1c	UEU1c	UEU1c	UEU1c	–	UEU1c
UUE1c	UEU1c	UUE1c	UUE1c	UUE1c	–	UUE1c
EE02c	EUU2c	EE02c	EEE3c	EEE3c	EEE2c.a-bc	EE02c
E0E2c	EUU2c	EEE3c	E0E2c	EEE3c	EEE2c.a-bc	E0E2c
0EE2c	0UU1c	0EE2c	0EE2c	0EE2c	0EE1c	0EE2c
EEE2c.a-bc	EUU2c	EEE2c.a-bc	EEE2c.a-bc	EEE2c.a-bc	–	EEE2c.a-bc
EEE2c.b-ac	EUU1c	EEE2c.b-ac	EEE2c.b-ac	EEE2c.b-ac	EEE1c	EEE2c.b-ac
EEE2c.c-ab	EUU1c	EEE2c.c-ab	EEE2c.c-ab	EEE2c.c-ab	EEE1c	EEE2c.c-ab
EUU2c	EEE2c.a-bc	EUU2c	EUU2c	EUU2c	–	EUU2c
UEU2c	UUE1c	UEU2c	UEU2c	UEU2c	UEU1c	UEU2c
UUE2c	UEU1c	UUE2c	UUE2c	UUE2c	UUE1c	UUE2c
EEE3c	EUU2c	EEE3c	EEE3c	EEE3c	EEE2c.a-bc	EEE3c

Table EC.7 Possible transitions from j^{+x} to j^{-}.

	Transition via													
States	`110`	`101`	`011`	`200`	`020`	`002`	`211`	`121`	`112`	`220`	`202`	`022`	`222`	`000`
0000c	–	–	–	E001c	–	00E1c	–	–	–	–	–	–	–	0000c
0001c	–	–	–	–	–	–	–	–	–	–	–	–	–	0001c
E001c	–	–	–	E001c	–	–	–	–	–	–	E0E2c	–	–	0001c
0E01c	–	–	–	–	0E01c	–	–	–	–	EE02c	–	0EE2c	–	0001c
00E1c	–	–	–	–	–	00E1c	–	–	–	–	E0E2c	–	–	0001c
EE01c	–	–	–	E001c	0E01c	–	–	–	–	EE01c	–	–	EEE2c.c-ab	0001c
E0E1c	–	–	–	E001c	–	00E1c	–	–	–	–	E0E1c	–	–	0001c
0EE1c	–	–	–	–	0E01c	00E1c	–	–	–	EE02c	–	0EE1c	EEE2abc	0001c
EEE1c	–	–	–	E001c	0E01c	00E1c	–	–	–	EE01c	E0E1c	0EE1c	EEE1c	0001c
UU01c	UU01c	–	–	–	–	–	–	–	UUE2c	–	–	–	–	–
U0U1c	–	U0U1c	–	–	–	–	–	–	–	–	–	–	–	–
0UU1c	–	–	0UU1c	–	–	–	EUU2c	–	–	–	–	–	–	–
EUU1c	–	–	0UU1c	–	–	–	EUU1c	–	–	–	–	–	–	–
UEU1c	–	U0U1c	–	–	–	–	–	UEU1c	–	–	–	–	–	–
UUE1c	UU01c	–	–	–	–	–	–	–	UUE1c	–	–	–	–	–
EE02c	–	–	–	–	–	–	–	–	–	EE02c	–	–	EEE3c	–
E0E2c	–	–	–	–	–	–	–	–	–	–	E0E2c	–	–	–
0EE2c	–	–	–	–	–	–	–	–	–	–	–	0EE2c	EEE3c	–
EEE2c.a-bc	–	–	–	–	–	–	–	–	–	EE02c	E0E2c	–	EEE2abc	–
EEE2c.b-ac	–	–	–	–	–	–	–	–	–	EE02c	–	0EE2c	EEE2c.b-ac	–
EEE2c.c-ab	–	–	–	–	–	–	–	–	–	–	E0E2c	0EE2c	EEE2c.c-ab	–
EUU2c	–	–	–	–	–	–	EUU2c	–	–	–	–	–	–	–
UEU2c	–	–	–	–	–	–	–	UEU2c	–	–	–	–	–	–
UUE2c	–	–	–	–	–	–	–	–	UUE2c	–	–	–	–	–
EEE3c	–	–	–	–	–	–	–	–	–	–	–	–	EEE3c	–

An instance of the SPRP for a two-block parallel-aisle warehouse and the optimal picker tour are depicted in Figure EC.1. The associated state space is shown in Figure EC.2. Moreover, the highlighted o-d-path from state `0000c` at stage 1^- to state `0001c` at stage 4^- (drawn in red/thick) describes the optimal picker tour.

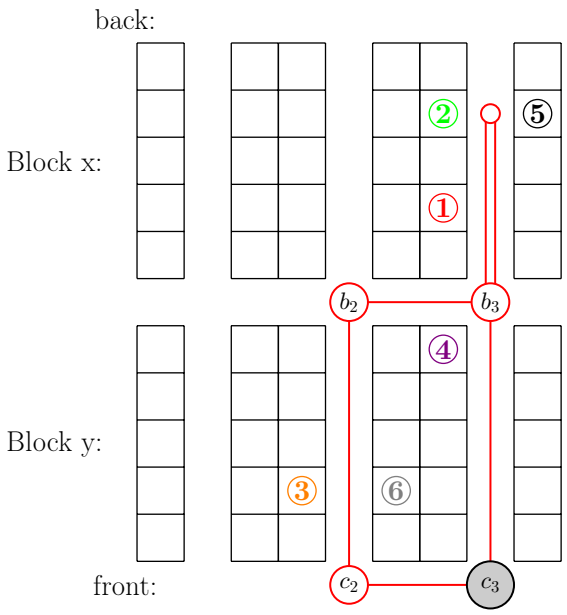

Figure EC.1 **Instance of the SPRP for a two-block parallel-aisle warehouse. The pick list contains six SKUs** $S = \{1, 2, 3, 4, 5, 6\}$ **(unit demand); pick operations are encircled.**

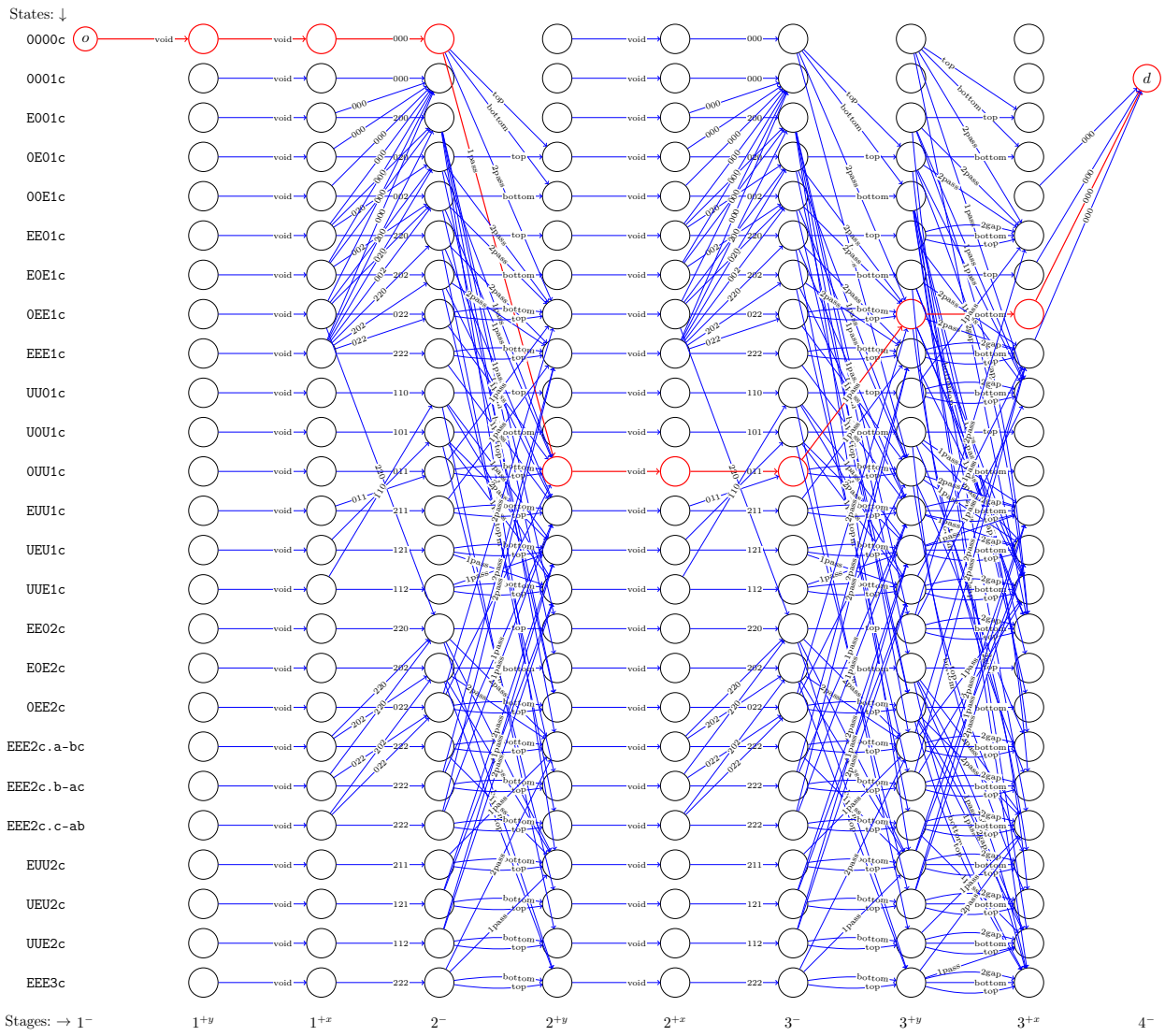

Figure EC.2 **State space (V, E) of the dynamic program for a two-block parallel-aisle warehouse and optimal sequence of states and actions (in red/thick).**

Appendix E: GTSP Formulation and Branch-and-Cut Algorithm

For describing the GTSP formulation and branch-and-cut algorithm of Fischetti et al. (2002), we make the following assumptions. An instance of the GTSP is given by an undirected graph (V, E) with vertex set V and edge set E. The vertices V are partitioned into m clusters $C_1, C_2, \ldots, C_m$, i.e., $V = \bigcup_{h \in H} C_h$ and $C_h \cap C_{h'} = \varnothing$ for all $h, h' \in H = \{1, 2, \ldots, m\}$ with $h \neq h'$. Let $h(i) \in H$ be the index of the cluster that contains $i \in V$, i.e., $i \in C_{h(i)}$. Edges $e = \{i, j\} \in e$ exist between every two i and j vertices with $h(i) \neq h(j)$. The cost of edge $e = \{i, j\} \in E$ is given by an integer number $c_e = c_{ij} = c_{ji}$. The GTSP is the problem of finding a cycle $T \subset E$ of minimal length $\sum_{e \in T} c_e$ such that T goes through every cluster at least once.

For $S \subseteq V$, the following sets and number are defined:

$$\begin{aligned} E(S) &= \{\{i, j\} \in E : i, j \in S\} \\ \delta(S) &= \{\{i, j\} \in E : i \in S, j \notin S \text{ or } i \notin S, j \in S\} \\ \mu(S) &= |\{h \in H : C_h \subseteq S\}| \end{aligned}$$

For $i \in V$, we write $\delta(i)$ as a shorthand notation for $\delta(\{i\})$.

Two types of decision variables are present in the GTSP formulation of Fischetti et al.: Binary variables x_e indicate whether edge $e \in E$ is present in the selected cycle and binary variables y_i indicate whether vertex $i \in V$ is included in the selected cycle. The binary formulation reads as follows:

$$z_{GTSP} = \sum_{e \in E} c_e \tag{EC.1a}$$

$$\text{subject to} \sum_{e \in \delta(i)} x_e - 2y_i = 0 \qquad \forall i \in V \tag{EC.1b}$$

$$\sum_{i \in C_h} y_i \geq 1 \qquad \forall h \in H \tag{EC.1c}$$

$$\sum_{e \in \delta(S)} x_e - 2y_i - 2y_j \geq -2 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2, i \in S, j \in V \setminus S \tag{GSEC}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{EC.1d}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{EC.1e}$$

The objective (EC.1a) is to minimize the length of the cycle. Constraints (EC.1b) couple the x- and y-variables guaranteeing that a selected vertex has degree two. Constraints (EC.1c) ensure that at least one vertex per cluster is visited. The *generalized subtour elimination constraints* (GSEC) forbid subtours over the selected vertices. Last, the domain of the variables is specified in (EC.1d) and (EC.1e).

There exist strengthened versions of the GSEC when complete clusters are included in S or its complement:

$$\sum_{e \in \delta(S)} x_e - 2y_i \geq 0 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2, i \in S, \mu(V \setminus S) > 0 \tag{GSEC'}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2, \mu(S) > 0, \mu(V \setminus S) > 0 \tag{GSEC"}$$

Our re-implementation of the branch-and-cut algorithm described by Fischetti et al. uses the `lazy` and `user` callback functions of the `C++` API of CPLEX.

- A method like `GSEC-BUILD` (Fischetti et al. 2002, p. 382) is used to find, for a given subset S, a most violated facet-defining inequality of type (GSEC), (GSEC'), or (GSEC"). The given set S is (heuristically) modified into S' in order to arrive at $\mu(S') > 0$ and/or $\mu(V \setminus S') > 0$ and to select better vertices i (and j).
- Integer solutions (x^*, y^*) (in the `lazy` cut callback) are inspected using a union-find algorithm (Tarjan 1975) to very quickly identify subsets S.
- For the separation of the strongest inequalities (GSEC"), we only rely on `GSEC-BUILD` and do not use a dedicated method like `GSEC-H1`, since we did not see a computational advantage in pre-tests.
- For fractional solutions (x^*, y^*), a spanning tree-based method like `GSEC-H2` (Fischetti et al. 2002, p. 382) is used as a fast heuristic separation procedure.
- Moreover, a method like `GSEC-SEP` (Fischetti et al. 2002, p. 382) is used (Fischetti et al. 2002, p. 383) for exact GSEC separation (to be precise, exact only if $\max_{i \in V} y_i^* = 1$, which is very often fulfilled).
- The overall strategy for calling the above heuristic and exact separation procedures follows a similar logic as described by Fischetti et al. However, we use a simplified management of selecting a subset of violated GSECs compared to the method `SEPARATION` (Fischetti et al. 2002, p. 386) (no loops with different minimum thresholds for violation), since the recent versions of CPLEX are excellent in handling cut pools itself.

References

Fischetti M, Salazar-Gonzalez JJ, Toth P (2002) The generalized traveling salesman and orienteering problems. Gutin G, Punnen A, eds., *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, chapter 13, 609–662 (Dordrecht: Kluwer), URL http://dx.doi.org/10.1007/0-306-48213-4_13.

Goeke D, Schneider M (2021) Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing* 33(2):436–451, URL http://dx.doi.org/10.1287/ijoc.2020.1040.

Ratliff HD, Rosenthal AS (1983) Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research* 31(3):507–521, URL http://dx.doi.org/10.1287/opre.31.3.507.

Roodbergen KJ, de Koster R (2001b) Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research* 133(1):32–43, URL http://dx.doi.org/10.1016/s0377-2217(00)00177-6.

Tarjan RE (1975) Efficiency of a good but not linear set union algorithm. *Journal of the ACM* 22(2):215–225, URL http://dx.doi.org/10.1145/321879.321884.

Weidinger F (2018) Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research* 95:139–150, URL http://dx.doi.org/10.1016/j.cor.2018.03.012.