Last update: 22.09.07

**Errata Brucker/Knust: Complex Scheduling, Springer, 2006**

- p.25, 1: Given $n$ **positive** integers ...
- p.26, 1: $C_i + p_j \leq C_j$
- p.30, Fig. 2.2, Step 7: `FOR ALL` $j \in \overline{S}$
- p.37, Fig. 2.9, Step 2: `FOR ALL` $(i,j) \in V \times V$ `with` $i \neq j$
- p.39, 3: maximizes
- p.43, 10: $\binom{n+m}{m}$
- p.52, (2.40): $\sum\limits_{j=1}^{n} r_{jk} \sum\limits_{\tau=t+1}^{\min\{t+p_j,T\}} x_{j\tau} \leq R_k \ (k = 1, \ldots, r; t = 0, \ldots, T)$
- p.53, -9: a typical value of $m$ is no more than 50
- p.54, -5: If $a_3 = 2$, then (2.47) implies $a_1 = 0$
- p.55, (2.50): $\sum\limits_{i=1}^{m} w_i a_i \leq W \ldots$
- p.59, Figure 2.12(d): $c_{ij}^r$ instead of $c_{ij}$
- p.62, Theorem 2.7: The proof should read "Define $x^1 := x - x^0$. If $x_{ij}^1 = x_{ij} - x_{ij}^0 > 0$, then $0 \leq x_{ij}^0 < x_{ij} \leq u_{ij}$. Thus, the arc $(i,j)$ belongs to $G(x^0)$ and we consider $x_{ij}^1$ as a flow on this arc in $G(x^0)$. On the other hand, if $x_{ij}^1 = x_{ij} - x_{ij}^0 < 0$, then $0 \leq x_{ij} < x_{ij}^0$. Thus, the arc $(j,i)$ belongs to $G(x^0)$ and we consider $-x_{ij}^1$ as a flow on this arc in $G(x^0)$. ..."
- p.65, 12: $u[X, \overline{X}] = \ldots$
- p.65, -10: (for node $j = p$)
- p.71, 18: the algorithm performs $2 \cdot 2^n$ augmentations
- p.78, Fig.2.23 should read:

```
Algorithm Branch-and-Bound Knapsack
 1.  L := 0;  k := 0;
 2.  FOR j := k + 1 TO n DO
 3.        x_j := ⌊(b − Σ_{i=1}^{j−1} a_i x_i)/a_j⌋;
 4.  IF Σ_{i=1}^{n} c_i x_i > L THEN
 5.        L := Σ_{i=1}^{n} c_i x_i;  x* := x
 6.  ENDIF;
 7.  l := n − 1;
 8.  IF no 1 ≤ k ≤ l with x_k > 0 exists THEN STOP;
 9.  Determine the largest 1 ≤ k ≤ l with x_k > 0;
10.  x_k := x_k − 1;
11.  IF Σ_{i=1}^{k} c_i x_i + (c_{k+1}/a_{k+1}) (b − Σ_{i=1}^{k} a_i x_i) < L + 1 THEN
12.        l := k − 1;
13.        GOTO 8;
14.  ENDIF;
15.  ELSE GOTO 2;
```

- p.81, 19: where initially $c(j,k) := -\infty$ for $j = 0, \ldots, n$ and $k < 0$
- p.82, 3: $F(S,j) = \min_{i \in S \setminus \{j\}} \{F(S \setminus \{j\}, i) + c_{ij}\}$
- p.102, (3.16): $P(\Omega_{l,k} \setminus \{i\})$
- p.102, -12: The definitions of $\Delta_{\lambda k}$ and $H_{ik}$ have to be

$$\Delta_{\lambda,k} := \max_{\nu \in \Omega_{\lambda,k}} \{r_\nu + P_{\nu,k}\}, \text{ and } H_{i,k} := \max_{\substack{\nu < i \\ d_\nu \leq d_k}} \{r_\nu + P_{\nu,k}\}.$$

- p.103, -3: The recursion should read $P_{n+1,k} = \Delta_{n+1,k} = H_{1k} := 0$,

$$P_{\lambda,k} = \sum_{\substack{\mu \geq \lambda \\ d_\mu \leq d_k}} p_\mu = \begin{cases} P_{\lambda+1,k} + p_\lambda, & \text{if } d_\lambda \leq d_k \\ P_{\lambda+1,k}, & \text{otherwise} \end{cases} \quad \text{for } \lambda = n, \ldots, 1,$$

$$\Delta_{\lambda,k} := \max_{\substack{\nu \geq \lambda \\ d_\nu \leq d_k}} \{r_\nu + P_{\nu,k}\} = \begin{cases} \max\{\Delta_{\lambda+1,k},\, r_\lambda + P_{\lambda,k}\}, & \text{if } d_\lambda \leq d_k \\ \Delta_{\lambda+1,k}, & \text{otherwise} \end{cases}$$

for $\lambda = n, \ldots, 1$, and

$$H_{\lambda,k} := \max_{\substack{\nu < \lambda \\ d_\nu \leq d_k}} \{r_\nu + P_{\nu,k}\} = \begin{cases} \max\{H_{\lambda-1,k},\, r_{\lambda-1} + P_{\lambda-1,k}\}, & \text{if } d_{\lambda-1} \leq d_k \\ H_{\lambda-1,k}, & \text{otherwise} \end{cases}$$

for $\lambda = 2, \ldots, n$

- p.106, (3.26): $P(\Omega_{j,l} \setminus \{i\})$
- p.108, 2: The recursion should read:
  Starting with $P'_{j0} := 0$, the values

$$P'_{j,\lambda} = \sum_{\substack{\mu \leq \lambda \\ r_\mu + p_\mu \geq r_j + p_j}} p_\mu = \begin{cases} P'_{j,\lambda-1} + p_\lambda, & \text{if } r_\lambda + p_\lambda \geq r_j + p_j \\ P'_{j,\lambda-1}, & \text{otherwise} \end{cases}$$

  for $\lambda = 1, \ldots, n$ can be computed in constant time from the previous values $P'_{j,\lambda-1}$. Furthermore, with

$$P_{j,\lambda} = \begin{cases} -\infty, & \lambda < j \\ P'_{j,\lambda}, & \lambda \geq j \end{cases}$$

  and the initial values $\Delta_{j,0} := \infty$ we calculate the values $\Delta_{j,\lambda}$.
- p.109: The $O(n^2)$-description of the input-or-output test is wrong. It can be repaired, but then an $O(n^3)$-algorithm is obtained.
- p.114, 1: For example, if the hypothetical constraint $S_i > t_i$ with $t_i \in [r_i, d_i - p_i[$ leads to a contradiction, the time window of activity $i$ can be reduced to $[r_i, t_i + p_i]$.
- p.121, 5: $\frac{UB-LB}{LB}$
- p.132, 8: upper bound (instead of "lower bound")
- p.144, -16: $[\max_{k=1}^{r}\{t^{m_k-1}\}, T]$
- Section 3.4.2: The algorithms are described in such a way that $R_k(t)$ denotes the capacity of resource $k$ in the interval $[t, t+1[$.
- p.148, Fig. 3.38: It would be more precise to replace steps 2 and 8 by:

```
2.  Let E_1 be the set of all activities i without predecessor
    and r_ik ≤ R_k(τ) for k = 1,...,r and all τ ∈ {0,...,p_i − 1};
8.          Add j to A_λ and update the set E_λ by eliminating
            j and all activities i with r_ik > R_k(τ) for some
            resource k and a τ ∈ {t_λ, t_λ + p_i − 1};
```

- p.149: Replace all $i$ by $j$ in the proof of Theorem 3.5.
- p.152, -1: We have at most $n(n-1)/2$ feasible left shift operators and at most $n(n-1)/2$ feasible right shift operators for a permutation.
- p.159: In the procedure in Fig.3.49 we have to distinguish between the set of eligible activities $E_\lambda$ and the set of branching candidates $B_\lambda$:

```
Procedure B&B (λ, F_λ, E_λ, S, s_{λ-1})
 1.  B_λ := E_λ;
 2.  WHILE B_λ ≠ ∅ DO
 3.     Choose an activity j ∈ B_λ;   B_λ := B_λ \ {j};
 4.     Calculate the smallest time t ≥ s_{λ-1} such that j
        can be scheduled in the interval [t, t + p_j[ without
        violating resource or precedence constraints;
 5.     Schedule j in the interval [t, t + p_j[ and set S_j := t;
 6.     F_λ := F_λ ∪ {j};   E_λ := E_λ \ {j};   s_λ := t;
 7.     IF λ = n + 1 THEN
 8.        UB := min {UB, S_{n+1}}; unschedule n + 1;
 9.        RETURN
10.     ELSE
11.        Let E_{λ+1} := E_λ and add to E_{λ+1} all successors
           i ∉ E_λ of j for which all predecessors are in F_λ;
12.        Calculate a lower bound LB(S) for all extensions
           of the current partial schedule S;
13.        IF LB(S) < UB THEN
14.           B&B (λ + 1, F_λ, E_{λ+1}, S, s_λ);
15.        Unschedule j and set F_λ := F_λ \ {j};   E_λ := E_λ ∪ {j};
16.     ENDIF
17.  ENDWHILE
```

- p.160, 15: The swapping rule, which can be applied for regular objective functions, should more precisely be formulated as:

  **Swapping rule:** If two activities $j, i$ are started at the same time, they can be swapped in the sequence without resulting in a worse schedule. Thus, if we have considered all extensions of $\ldots, i, j$ we do not have to consider the extensions of $\ldots, j, i$. Hence, in this situation it is sufficient to consider only extensions $\ldots, i, j$ with $i < j$.
- p.164, Fig. 3.51: In the left box for $\lambda = 2$ it has to be $A_2 = \{1\}$. In the box above schedule $S_4$ activities 2 and 3 should start at the same time.
- p.182, 13: $(b_k - y)^+ = \min\{\alpha_k \mid \alpha_k \geq 0, y + \alpha_k \geq b_k\}$ for $k = 1, \ldots, l$ and $(y - b_{k-1})^+ = \min\{\alpha_k \mid \alpha_k \geq 0, -y + \alpha_k \geq -b_{k-1}\}$ for $k = l+1, \ldots, m+1$