Algorithm CNR

Input: A state $q \in Q_R(q_0)$, D_1, D_2, D_3, and S;
Output: Φ; /* Φ stores the number of required raw parts for all part types */

Set $q(\lambda) = 1$;
for (j = 1: n) **do**
    $\Phi[j] = 0$;
end for
for (i = 1: m) **do** /*set the capacity of resources to the infinity*/
    ψ_i = INF;
end for
for (i = 1: n) **do** /*clear all parts not at stages in D_1 to avoid disturbance */
    for (j = 1 : l_i) **do**
        if ($P_{ij} \notin D_1$)
            $q(x_{ij}) = 0$;
            $q(y_{ij}) = 0$;
        else /*suppose that all parts have finished their current operations*/
            $q(x_{ij}) = q(x_{ij}) + q(y_{ij})$;
            $q(y_{ij}) = 0$;
        end if-else
    end for
end for
for (i = 1 : $|D_1|$) **do** /*advance those parts that can be advanced to their proper positions*/
    Suppose $D_1[i] = P_{jk}$, $D_3[i] = P_{vw}$;
    while ($D_2[i] > 0$) **do**
        if (the part at stage $D_1[i]$ can be advanced to $D_3[i]$)
            Let ζ be the sequence of stages which are gone through from $D_1[i]$ to $D_3[i]$;
            Advance a part at stage $D_1[i]$ to $D_3[i]$;
            for (t = 1: $\mathbb{c}$) **do** /*update q, D_1 and D_2*/
                let $\Xi = P_{at} \cap \zeta$;
                for ($P_{ef} \in \Xi$) **do**
                    for ($P_{gh} \in \{\Pi(P_{ef})\backslash\zeta\}$) **do**
                        $q(x_{gh}) = q(x_{gh}) - 1$;
                        find $d \in \{1, 2, \ldots, |D_1|\} \ni P_{gh} = D_1[d]$;
                        $D_2[d] = D_2[d] - 1$;
                        if ($D_2[d] = 0$)
                            set $D_1[d] = P_{vw}$;
                        end if
                    end for
                end for
                $q(x_{jk}) = q(x_{jk}) - 1$ and $q(x_{vw}) = q(x_{vw}) + 1$;
                $D_2[i] = D_2[i] - 1$;
                if ($D_2[i] = 0$)
                    $D_1[i] = P_{vw}$;
                end if
            end for
        else
            find $c \in \{1, 2, \ldots, l_j - k\}$ such that $P_{j(k+c)} \in \Re_{jk}$, $P_{j(k+c)} \in \Pi_a$ and $P_{j(k+w)} \notin \Pi_a$ $\forall w \in \{1, 2, \ldots c - 1\}$;
            $D_1[i] = P_{j(k+c)}$;
            $q(x_{jk}) = q(x_{jk}) - D_2[i]$ and $q(x_{j(k+c)}) = q(x_{j(k+c)}) + D_2[i]$;
            break;
        end if-else
    end while
end for
for (i = 1: $\mathbb{c}$) **do** /* starts the computation from the first class assembly operations */
    for ($P_{jk} \in P_{ai}$) **do**
        for (t = 1 : $|D_1|$) **do**
            if ($D_1[t] \in \Pi(P_{jk})$)
                Let $z = \max\{q(x_{vw}) + q(y_{vw}) : P_{vw} \in \Pi(P_{jk})\}$;
                for ($P_{vw} \in \Pi(P_{jk})$) **do**

CNR (continued)

Let $\varsigma(P_{vw}) = \Theta(P_{vw}) \cap P_b$;
for($P_{ob} \in \varsigma(P_{vw})$) **do**
  Let $\Phi[o] = \Phi[o] + z - q(x_{vw}) - q(y_{vw})$;
end for
$q(x_{vw}) = 0$;
for ($f = 1 : |D_1|$) **do**
  if ($f \neq t$ and $D_1[f] \in \Pi(P_{jk})$)
    $D_2[f] = 0$;
    $D_1[f] = D_3[f]$;
  end if
end for
end for
$q(x_{jk}) = q(x_{jk}) + z$; /* assemble the parts at stages in $\Pi(P_{jk})$ z times *
$D_1[t] = P_{jk}$;
$D_2[t] = z$;
while ($D_2[t] > 0$) **do**
  Suppose $D_3[t] = P_{vw}$;
  if (a part at stage $D_1[t]$ can be advanced to $D_3[t]$)
    /*advance the obtained parts one by one to $D_3[t]$ */
    Let ζ be the sequence of stages which are gone through from $D_1[t]$ to $D_3[t]$;
    Advance a part at stage $D_1[t]$ to $D_3[t]$;
    for ($d = 1$: $\mathbb{C}$) **do** /*update q, D_1 and D_2*/
      let $\Xi = \{P_{ad} \cap \zeta\}$;
      for ($P_{ef} \in \Xi$) **do**
        for ($P_{gh} \in \{\Pi(P_{ef}) \backslash \zeta\}$) **do**
          $q(x_{gh}) = q(x_{gh}) - 1$;
          find $o \in \{1, 2, \ldots, |D_1|\} \ni P_{gh} = D_1[o]$;
          $D_2[o] = D_2[o] - 1$;
          if ($D_2[o] = 0$)
            $D_1[o] = D_3[t]$;
          end if
        end for
      end for
    end for
    $q(x_{jk}) = q(x_{jk}) - 1$ and $q(x_{vw}) = q(x_{vw}) + 1$;
    $D_2[t] = D_2[t] - 1$;
    if ($D_2[t] = 0$)
      $D_1[t] = D_3[t]$;
    end if
  else
    find $c \in \{1, 2, \ldots, l_j - k\}$ such that $P_{j(k+c)} \in \Re_{jk}$, $P_{j(k+c)} \in \Pi_a$ and $P_{j(k+w)} \notin \Pi_a$ $\forall w \in \{1, 2, \ldots c - 1\}$;
    $D_1[t] = P_{j(k+c)}$;
    $q(x_{jk}) = q(x_{jk}) - D_2[t]$;
    $q(x_{j(k+c)}) = q(x_{j(k+c)}) + D_2[t]$;
    break;
  end if-else
end while
end if
end for
end for
end for
return Φ;
End

Theorem 1: The complexity of Algorithm CNR is polynomial.

Proof: The complexity of Lines 2-4 is $O(n)$, that of Lines 5-7 is $O(m)$, that of Lines 8-18 is $O(L \times |D_1|)$, where $L = l_1 + l_2 + \ldots + l_n$. The for-loop from Line 19 to Line 50 can execute no more than $|D_1|$. Since a part occupies a buffer slot and all part at the same stage occupy the buffer slots of the same resource, $\forall i \in \{1, 2, \ldots, |D_1|\}$, $D_2[i] \le \psi_{\max} = \max\{\psi_j \mid j \in \{1, 2, \ldots, m\}\}$. The while-loop from Line 21 to Line 49 can execute no more than $\psi_{\max}$ times. Since the number of stages between stage $D_1[i]$ to $D_3[i]$ is no more than $l_{\max} = \max\{l_j \mid j \in \{1, 2, \ldots, n\}\}$, the complexity of advancing a part at stage $D_1[i]$ to $D_3[i]$ is bound by $l_{\max}$. Since there are at most L stages needing to be updated, the complexity of the for-loop from Line 25 to Line 42 is bounded by L. The complexity of Lines 44 to Line 47 is dominated by that of Line 44, which is bound by $O(l_{\max})$. Thus, the complexity of Lines 19-50 is $O(|D_1| \times \psi_{\max} \times (l_{\max} + L + l_{\max})) = O(|D_1| \times \psi_{\max} \times L)$.

The for-loop from Line 53 to Line 104 the can execute $|P_{a1}| \times |D_1| + |P_{a2}| \times |D_1| + \ldots + |P_{a\mathbb{c}}| \times |D_1| = |P_a| \times |D_1|$ times. Since $|\Pi(P_{jk})| \le L$, the complexity of Line 55 is $O(L)$ and the for-loop from 56-68 can execute no more than L times. Since $|\varsigma(P_{vw})| \le n$, The complexity of Lines 58-60 is $O(n)$. The complexity of Lines 62-67 is $O(|D_1|)$. Thus, the complexity of the for-loop from 56-68 is $O(L \times (n + |D_1|))$. Since a part occupies a buffer slot and all part at the same stage occupy the buffer of the same resource, $\forall i \in \{1, 2, \ldots, |D_1|\}$, $D_2[i] \le \psi_{\max}$. The while-loop from Line 72 to Line 102 can execute no more than $\psi_{\max}$ times. Since the number of stages between stage $D_1[t]$ to $D_3[t]$ is no more than $l_{\max}$, the complexity of advancing a part at stage $D_1[t]$ to $D_3[t]$ is bound by $l_{\max}$. The for-loop from Lines 77 to Line 89 updates q, D_1, and D_2. As stated before, its complexity is bounded by L. The complexity of Lines 96 to Line 99 is dominated by that of Line 96, which is bound by $O(l_{\max})$. Thus, the complexity of the for-loop from Line 51 to Line 106 is $O(|P_a| \times |D_1| \times (L \times (n + |D_1|) + \psi_{\max} \times (l_{\max} + L))) = O(|P_a| \times |D_1| \times L \times (n + |D_1| + \psi_{\max}))$.

Thus, $O(\text{CNR}) = O(n) + O(m) + O(L \times |D_1|) + O(|D_1| \times \psi_{\max} \times L) + O(|P_a| \times |D_1| \times L \times (n + |D_1| + \psi_{\max}))$. Since $L \ge n$ and $L \ge m$, $O(\text{CNR}) = O(m) + O(|P_a| \times |D_1| \times L \times (n + |D_1| + \psi_{\max})) = O(|P_a| \times |D_1| \times L \times (n + |D_1| + \psi_{\max}))$. ■

Theorem 2: Given a state $q \in Q_R(q_0)$, D_1, D_2, D_3 and S, let $\Phi_{\min}$ be the minimum number of raw parts required to advance all parts at stages in D_1 to their proper stages in D_3. Then, $\Phi = \text{CNR}(q, D_1, D_2, D_3, S) = \Phi_{\min}$.

Proof: We prove that $\forall \mathbb{c} \in Z$, $\Phi = \Phi_{\min}$ by mathematical induction on the value of $\mathbb{c}$ as follows.

If $\mathbb{c} = 0$, then no assemble operation exist in S. All parts at stages in D_1 can be advanced to their proper stages in D_3 without needing any raw parts. Thus, $\forall i \in \{1, 2, \ldots, n\}$, $\Phi[i] = \Phi_{\min}[i] = 0$.

If $\mathbb{c} = 1$, after the execution of Lines 1-50, all part remaining in the system stay in $\Pi(P_{jk})$, where $P_{jk} \in P_{a1}$. In order to advance all parts at stages in $P_{vw} \in \Pi(P_{jk})$ to their proper stages, stage P_{jk} must be gone through. Thus, for stage $P_{vw} \in \Pi(P_{jk})$, at least $z = \max\{q(x_{vw}) + q(y_{vw}): P_{vw} \in \Pi(P_{jk})\}$ parts are needed. Since $\forall P_{vw} \in \Pi(P_{jk})$, no part stays in $\{\theta(P_{vw}) \setminus P_{vb}\}$, the lacking parts can only be obtained from the fictitious beginning places. Thus, $\forall i \in \{1, 2, \ldots, n\}$, $\Phi[i] = \Phi_{\min}[i]$.

Suppose that $\forall i \in \{1, 2, \ldots, n\}$, $\Phi[i] = \Phi_{\min}[i]$ when $\mathbb{c} = h \in Z^+$. Now, we prove $\forall i \in \{1, 2, \ldots, n\}$, $\Phi[i] = \Phi_{\min}[i]$ when $\mathbb{c} = h + 1$. Note that CNR computes the number of raw parts from smaller assembly class to larger assembly class and the part obtained after the assembly operation is advanced either to their proper stages (Lines 74-94) or to the next stages in Π_a (Lines 95-101). In other words, before CNR computing the number of raw parts for $P_{jk} \in P_{a(h+1)}$, all parts remaining in the system stay in $\Pi(P_{jk})$. In order to advance all parts at stages in $P_{vw} \in \Pi(P_{jk})$ to their proper stages, stage P_{jk} must be gone through. Thus, for stage $P_{vw} \in \Pi(P_{jk})$, at least $z = \max\{q(x_{vw}) + q(y_{vw}) : P_{vw} \in \Pi(P_{jk})\}$ parts are needed. Since $\forall P_{vw} \in \Pi(P_{jk})$, no part stays in $\{\theta(P_{vw}) \setminus P_{vb}\}$, the lacking parts can only be obtained from the fictitious beginning places. Thus, the number of raw parts which is worked out by CNR for $P_{a(h+1)}$ is minimal. By the assumption, the number of raw parts which is worked out by CNR for P_{ai} $\forall i \in \{1, 2, \ldots, h\}$ is minimal. Thus, $\forall i \in \{1, 2, \ldots, n\}$, $\Phi[i] = \Phi_{\min}[i]$ holds for $\mathbb{c} = h + 1$. ■

AMSs in Figs. 1, 2, and 3 are used to test the runtime of MBA_1 and MBA_2. They are implemented in C++ and run on a 3.4 GHz desktop computer with 16G RAM. Its operating system is Windows 7 Professional. Simulation results are shown in Table I. From it, we know that MBA_1 and MBA_2 can averagely detect the safety of states of tested AMSs in 0.452 μs and 10.944 μs, respectively. Besides, from Table I, we find that $AR(MBA_1) \approx 0.00013 \times \psi_{sum} \times L^2$ μs and $AR(MBA_2) \approx 0.00014 \times \psi_{sum}^2 \times L^2$ μs. It means that MBA_1 can detect the safety of states of a system with $L = 2500$ and $\psi_{sum} = 1200$ in 1 s averagely and MBA_2 can detect the safety of states of a system with $L = 420$ and $\psi_{sum} = 200$ in 1 s averagely. Thus, MBA_1 and MBA_2 are capable of handling large scale AMSs.

TABLE I

SIMULATION RESULTS OF MBA_1 AND MBA_2.

	S in Fig. 1 n=4, ψ_{sum}=9, l_{max}=5, L=18		*S* in Fig. 2 n=5, ψ_{sum}=5, l_{max}=2, L=9		*S* in Fig. 3 n=4, ψ_{sum}=24, l_{max}=3, L=12	
method	MBA_1	MBA_2	MBA_1	MBA_2	MBA_1	MBA_2
NTS	1000	1000	1000	1000	1000	1000
TR (μs)	398	3639	55	290	452	11434
AR (μs)	0.398	3.639	0.055	0.29	0.452	11.434

NTS is short for the number of tested states; TR is short for total runtime; AR is short for average runtime.