Algorithm 5 Constructing a path.

Input: $Closed$, S_0, S_G.
Output: A path from S_0 to S_G.

function CONSTRUCT($Closed$, S_0, S_G)
  $path := \emptyset$; $a := 0$; $S' := S_G$; $\overleftarrow{g} := 0$;
  while $S' \neq S_0$ **do**
    $S := S'$;
    for all $t \in \overleftarrow{E_t}(S)$ **do**
      if $(S', c) :=$ BackT($Closed, S_0, S, t) \neq \emptyset$ **then**
        $path[a] := (S_G.f - \overleftarrow{g}, t)$;
        $\overleftarrow{g} = \overleftarrow{g} + c$;
        $a := a + 1$;
        break;
      end if
    end for
  end while
  return $path$;
end function

Bo Huang (M'14-SM'15) received his B.S. and Ph.D. degrees from Nanjing University of Science and Technology (NUST), Nanjing, China, in 2002 and 2006, respectively. He joined NUST in 2007, where he is currently a full Professor at the School of Computer Science and Engineering of NUST. He has been a Visiting Scholar at the University of Missouri - Kansas City, MO, USA, in 2013, and at the New Jersey Institute of Technology, Newark, NJ, USA, from 2014 to 2015 and from 2019 to 2020, respectively. His research interests include discrete event systems, Petri nets, intelligent manufacturing, transportation and robotic systems. He has taken charge of more than 10 projects and published more than 60 papers in the above areas. He has been serving as a Reviewer for more than 10 international journals and served as a Session Chair or a PC member for more than 10 academic conferences.

Algorithm 6 Finding an immediate predecessor of S in *Closed* (for nets with $\lambda = 2$).

Input: $Closed$, S_0, S, t that is reversely enabled at S.
Output: (S', c) with $S' \in$ *Closed*, $S'[t\rangle S$, $S.g = S'.g + c$ and $c = c(S', S)$; otherwise, $\emptyset$.

function BACKT($Closed, S_0, S, t$)
  /* Moves tokens for the reversely firing of t. */
  for all $p \in t^\bullet \cup {}^\bullet t$ **do**
    $S.M(p) := S.M(p) - [N](p, t)$;
  end for
  if $S.M = S_0.M$ **then**
    return $(S_0, 0)$;
  end if
  /* Handles the remaining times of tokens in the activity post-place of t. */
  if $\{q\} := \{p | p \in t^\bullet$ and $D(p) > \emptyset\} \neq \emptyset$ **then**
    if $S.R_1(q) = D(p)$ **then**
      $S.R_1(q) := 0$;
    else
      $S.R_2(q) := 0$;
    end if
  end if
  $S' := S$;
  for all $t' \in \overleftarrow{E_t}(S'.M)$ **do**
    $\mathcal{C} := \prod_{p_i \in P} [M(p_i) = S'.M(p_i)]$;
    $\{p'\} := t'^\bullet \cap P_A$;
    $\{p''\} := {}^\bullet t \cap (P \setminus P_R)$;
    $d := D(p') - \max\{S'.R_1(p'), S'.R_2(p')\}$;
    if $d > D(p'') - \max\{S'.R_1(p''), S'.R_2(p'')\}$ **then**
      continue;
    end if
    if $\max\{S'.R_i(p')\} > 0$ or $D(p') = 0$ or $p' = p''$ **then** /* If the time needed by $S'[t\rangle S$ equals d. */
      $\mathcal{C} := \mathcal{C} \cdot$ Constraints(S', p', p'', d);
    else
      $\mathcal{C}_1 :=$ bddfalse;
      for all k such that $0 \leq k \leq D(p'') - \max\{S'.R_1(p''), S'.R_2(p'')\} - d$ **do**
        $\mathcal{C}_1 := \mathcal{C}_1 +$ Constraints$(S', p', p'', d + k)$;
      end for
      $\mathcal{C} := \mathcal{C} \cdot \mathcal{C}_1$;
    end if
    /* Find a state that satisfies $\mathcal{C}$ in $Closed$. */
    $found := Closed \cdot \mathcal{C}$;
    if $found \neq$ bddfalse **then**
      $S' :=$ bdd2var(bdd_satone($found$)); /* Find a satisfying variable assignment. */
      if $S'.M(p'') = 1$ **then**
        return $(S', \max\{S'.R_1(p''), S'.R_2(p'')\})$;
      else /* $S'.M(p'') = 2$. */
        return $(S', \min\{S'.R_1(p''), S'.R_2(p'')\})$;
      end if
    end if
  end for
  return $\emptyset$; /* If no such a state in *Closed*. */
end function

Algorithm 7 Construct constraints of f and R_i for S'.

Input: S', p' the activity post-place of a transition t', p'' the non-resource pre-place of a transition t, and d the time required for p' to reversely enable t' at S' such that $\exists S, S''$, $S'[t\rangle S$ and $S''[t'\rangle S'$.

Output: Constraints $\mathcal{C}$.

function CONSTRAINTS(S', p', p'', d)
  /* Constraints of f. */
  $\{l\} := \{p | S'.M(p) > 0$ and $D(p) > 0$ and $p \in H_l(r_{\max})\}$;
  if $\{l\} = \emptyset$ **then**
    $\mathcal{C} := \mathcal{C} \cdot [f = (S'.f - d)]$;
  else if $l = p''$ **then**
    $\mathcal{C} := \mathcal{C} \cdot (f = S'.f)$;
  else
    if $S'.R_1(l) > 0$ or $S'.R_2(l) > 0$ **then**
      $\mathcal{C} := \mathcal{C} \cdot (f = S'.f)$;
    else /*$S'.R_1(l) = S'.R_2(l) = 0$*/
      $\mathcal{C}_1 :=$ bddfalse;
      for all a such that $0 \leq a \leq d$ **do**
        if $a = 0$ **then**
          $\mathcal{C}_1 := \mathcal{C}_1 + [f = (S'.f - d)] \cdot [R_1(l) = 0] \cdot [R_2(l) = 0]$;
        else
          $\mathcal{C}_1 := \mathcal{C}_1 + [f = (S'.f - d + a)] \cdot [R_1(l) = a + R_2(l) = a]$;
        end if
      end for
      $\mathcal{C} := \mathcal{C} \cdot \mathcal{C}_1$;
    end if
  end if
  /* Constraints of R in p' and p''. */
  $\mathcal{C} := \mathcal{C} \cdot [R_1(p') = D(p') + R_2(p') = D(p')]$;
  if $D(p'') > 0$ **then**
    if $p'' = p'$ **then**
      $\mathcal{C} := \mathcal{C} \cdot [R_1(p'') = (S'.R_1(p'') + d)] \cdot [R_2(p'') = (S'.R_2(p'') + d)]$;
    else
      if $S'.M(p'') = 1$ **then**
        if $d > 0$ **then**
          $\mathcal{C} := \mathcal{C} \cdot [R_1(p'') = d + R_2(p'') = d]$;
        else /* $d = 0$.*/
          $\mathcal{C} := \mathcal{C} \cdot [R_1(p'') = 0] \cdot [R_2(p'') = 0]$;
        end if
      else
        $\mathcal{C} := \mathcal{C} \cdot [R_1(p'') = (S'.R_1(p'') + d)] \cdot [R_2(p'') = (S'.R_2(p'') + d)]$;
      end if
    end if
  end if
  /* Constraints of R for the remaining activity places. */
  for all $q' \in \{p | S'.M(p) > 0$ and $D(p) > 0$ and $p \neq p'$ and $p \neq p''\}$ **do**
    if $S'.M(q') = 1$ **then**
      if $S'.R_1(q') > 0$ **then**
        $\mathcal{C} := \mathcal{C} \cdot [R_1(q') = (S'.R_1(q') + d) \cdot R_2(q') = 0]$;
      else if $S'.R_2(q') > 0$ **then**
        $\mathcal{C} := \mathcal{C} \cdot [R_2(q') = (S'.R_2(q') + d) \cdot R_1(q') = 0]$;
      else
        $\mathcal{C} := \mathcal{C} \cdot [R_1(q') \trianglelefteq d \cdot R_2(q') = 0 + R_2(q') \trianglelefteq d \cdot R_1(q') = 0]$;
      end if
    else /* $S'.M(q') = 2$. */
      if $S'.R_1(q') > 0$ **then**
        $\mathcal{C} := \mathcal{C} \cdot [R_1(q') = (S'.R_1(q') + d)]$;
      else
        $\mathcal{C} := \mathcal{C} \cdot [R_1(q') \trianglelefteq d]$;
      end if
      if $S'.R_2(q') > 0$ **then**
        $\mathcal{C} := \mathcal{C} \cdot [R_2(q') = (S'.R_2(q') + d)]$;
      else
        $\mathcal{C} := \mathcal{C} \cdot [R_2(q') \trianglelefteq d]$;
      end if
    end if
  end for
  return $\mathcal{C}$;
end function