1. TSS-based BFS for optimizing the makespan of AMSs is given in Algorithm 1.

Algorithm 1 TSS-based BFS for optimizing the makespan of AMSs

Input: An AMS and its PPN model.

Output: Optimal schedule α and $C_{max}(\alpha)$.

Initialize: OPEN[0] = $\{TM_0[\varepsilon]\}$, OPEN[i] = $\emptyset$, $i \in Z_K$, and FINAL = $\emptyset$. /* OPEN[i], $i \in Z_K$ is a list used for storing timed states generated in layer i when searching the TSS of an AMS, K is the maximum layer value in the TSS, and FINAL is a list used for storing final timed states. */

$k = 0$;

while(OPEN[k] $\neq \emptyset$)do{

  for($TM[\upsilon] \in$ OPEN[k]){

    Compute $\Delta(TM[\upsilon])$;/*$\Delta(TM[\upsilon])$ is a set of transitions that are enabled at $TM[\upsilon]$. */

    for($t \in \Delta(TM[\upsilon])$){

      Fire transition t, obtain $\upsilon_1 = \upsilon t$ and $TM_1[\upsilon_1]$;

      $\Delta(TM[\upsilon]) := \Delta(TM[\upsilon]) \backslash t$;

      if($TM_1[\upsilon_1]$ is a new final timed state){FINAL := FINAL$\cup TM_1[\upsilon_1]$;}

      else if(there exist a timed state $TM_f[\alpha]$ in FINAL satisfying $TM_f[\alpha] = TM_1[\upsilon_1]$){

        if($\tau_{|\alpha|} < \tau_{|\upsilon_1|}$){$TM_f[\alpha] := TM_1[\upsilon_1]$;}}

      else if(there exist a timed state $TM_2[\upsilon_2]$ in OPEN[k+1] satisfying $TM_2[\upsilon_2] = TM_1[\upsilon_1]$){

        if($\tau_{|\upsilon_2|} < \tau_{|\upsilon_1|}$){$TM_2[\upsilon_2] := TM_1[\upsilon_1]$;}}

      else{OPEN[k+1] := OPEN[k+1]$\cup TM_1[\upsilon_1]$;} /* $TM_1[\upsilon_1]$ is not in OPEN[k+1]. */

    }**end for**

  OPEN[k] := OPEN[k]$\backslash TM[\upsilon]$;

}**end for**

$k := k + 1$;

}**end while**

Output the best schedule in FINAL;

End

2. TSS-based A* for optimizing the makespan of AMSs is shown in Algorithm 2.

Algorithm 2 TSS-based A* for optimizing the makespan of AMSs

Input: An AMS and its PPN model.

Output: Optimal schedule α and $C_{max}(\alpha)$.

Initialize: OPEN = $\{TM_0[\varepsilon]\}$and CLOSED = $\emptyset$. /*OPEN is a list used for storing unexplored timed states, and CLOSED is a list used for storing explored timed states. */

$k = 0$;

while(OPEN $\neq \emptyset$)do{

  select the state $TM[\upsilon]$ with the smallest $f(TM[\upsilon])$;/*$f(TM[\upsilon])$ is the heuristic function.*/

  if($TM[\upsilon]$ is the final timed state){$\alpha := \upsilon$, $C_{max}(\alpha) = \tau_{|\upsilon|}$, break;}

  else{

    Compute $\Delta(TM[\upsilon])$;/*$\Delta(TM[\upsilon])$ is a set of transitions that are enabled at $TM[\upsilon]$. */

    for($t \in \Delta(TM[\upsilon])$){

      Fire transition t, obtain $\upsilon_1 = \upsilon t$ and $TM_1[\upsilon_1]$;

```
        Δ(TM[υ]) := Δ(TM[υ])\t;
        if(there exist a timed state TM2[υ2] in OPEN satisfying TM2[υ2] = TM1[υ1]){
            if(τ|υ2| < τ|υ1|){TM2[υ2] := TM1[υ1];}}
        else if(there exist a timed state TM3[υ3] in CLOSED satisfying TM3[υ3] = TM1[υ1]){
            if(τ|υ3| < τ|υ1|){CLOSED := CLOSED\TM3[υ3]; OPEN := OPEN∪TM3[υ3];}}
        else{OPEN := OPEN∪TM1[υ1];}/* TM1[υ1] is neither in OPEN nor CLOSED. */
    }end for
    OPEN := OPEN\TM[υ];
    CLOSED := CLOSED∪TM[υ];
}end while
Output α and Cmax(α);
End
```

3. The implementation of TSS-based A* method is illustrated in Example 1.

Example 1: Consider the AMS in Example 3. Let the heuristic function in A* be $f(TM[\upsilon]) = \tau_{|\upsilon|}(\upsilon)$, i.e., the actual time cost function of $TM[\upsilon]$, which is admissible. Fig. 1 records the first four search steps of TSS-based A* on the considered AMS.

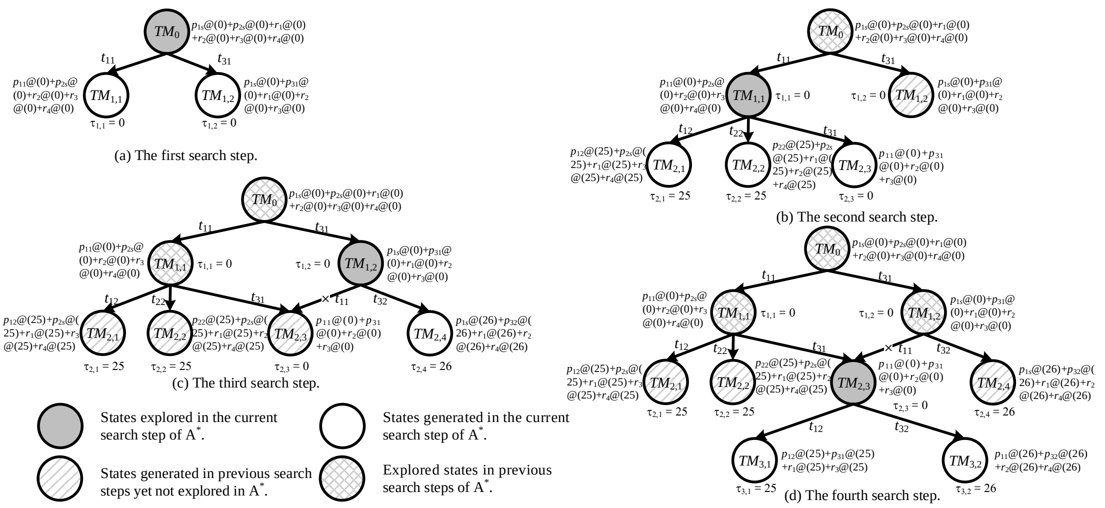

Fig. 1. The first four search steps of TSS-based A*.

As shown in Fig. 1, the initial timed state $TM_0[\varepsilon] = p_{1s}@(0) + p_{2s}@(0) + r_1@(0) + r_2@(0) + r_3@(0) + r_4@(0)$ is searched in the first search step of TSS-based A*, and two new states $TM_{1,1}[\upsilon_{1,1}]$ and $TM_{1,2}[\upsilon_{1,2}]$ are generated. Since the heuristic function values of $TM_{1,1}[\upsilon_{1,1}]$ and $TM_{1,2}[\upsilon_{1,2}]$ are equal, i.e., $f(TM_{1,1}[\upsilon_{1,1}]) = f(TM_{1,2}[\upsilon_{1,2}]) = 0$, randomly select one for further exploration. In the second search step, $TM_{1,1}[\upsilon_{1,1}]$ is selected, and by separately firing the enabled transitions, three new states $TM_{2,1}[\upsilon_{2,1}]$, $TM_{2,2}[\upsilon_{2,2}]$, and $TM_{2,3}[\upsilon_{2,3}]$ can be generated in this step. Similarly, in the third and fourth search steps, the unexplored state with the best heuristic function value is selected for exploration, which are $TM_{1,2}[\upsilon_{1,2}]$ and $TM_{2,3}[\upsilon_{2,3}]$, respectively. In Fig. 1, the explored states, generated states, and generated but not explored states in the first four search steps of TSS-based A* are presented. ♣

4. Time complexity of HHS algorithm

Given an AMS and its PPN model, suppose the total number of jobs to be processed is u, the capacity of each type of resources is $C(r)$, $\forall\, r \in R$, and the number of places and transitions in the PPN model are $|P|$ and $|T|$, respectively. The number of transitions on the longest place-transition path in the PPN model is L_m.

HHS consists of multiple iterations, and the main search process (lines 4-27) is executed once in each iteration. Since the value of W varies according to the number of iterations, and the actual number of iterations is related to the time complexity of the main search process, we only need to analyze the time complexity of the main search process. The main search process includes four parts: the applying of the hybrid search strategy, the exploration of the best W_a states in the current layer (OPEN[i], $i \in [0, K\text{-}1]$), the duplicate detection of the newly generated states, and the evaluation and pruning of the states of the next layer, i.e., OPEN[i+1]. The time complexity of each part is analyzed as follows.

Analysis: In the *iter*-th iteration, the actual number of states to be explored in the i-th ($i \in [0, K\text{-}1]$) search step is W_a and $W_a \le W$, which is determined by Algorithm 2 (Line 5 of Algorithm 3). The time complexity of Algorithm 2 is $O([2W+1]\log[2W+1]) + O(W)$. In detail, the number of states in OPEN[i] and CLOSED[i] is less than W+1 and W (determined by the pruning of states), respectively, so the time complexity of sorting the states in OPEN[i] and CLOSED[i] (line 5 of Algorithm 2) is $O([2W+1]\log[2W+1])$, and the comparison and deletion of states (line 7 of Algorithm 2) is $O(W)$.

In each search step, the time complexity of exploring the W_a states is $O(W_a|T|[|P|+|P|\cdot|T|])$. In detail, given a state, $|T|$ transitions should be checked and the resulting states should be detected, the corresponding time complexity is $O(|T|\cdot|P|)$. After firing feasible transitions at a state, at most $|T|$ states can be obtained, and the complexity for detecting the safety of the states (line 13 of Algorithm 3) is $O(|T|\cdot[|P|\cdot|T|])$.

Since at most $W_a|T|$ states can be generated in the i-th search step, the time complexity for duplicate detection of these states is $O(W_a|T|[u+W_a|T|\cdot|P|+W|P|])$. That is, each state should be evaluated first (line 14 of Algorithm 3), and the time complexity is $O(u)$. Then, the duplicated detection process is performed between the state and the states in OPEN[i+1] and CLOSED[i+1]. Since the number of states in OPEN[i+1] and CLOSED[i+1] is less than $W_a|T|$ and W, respectively, the time complexity of the duplicated detection process is $O([W_a|T| + W]|P|)$ in the worst case. Therefore, the time complexity of the duplicated detection of all generated states is $O(W_a|T|[u+W_a|T|\cdot|P|+W|P|])$.

In the phase of evaluating and pruning states (line 26 of Algorithm 3), the states in OPEN[i+1] are evaluated and sorted, the corresponding time complexity is $O(W_a|T|\cdot|P|C_r) + O(W_a|T|\log[W_a|T|])$, where $C_r = max\{C(r) \mid r \in R\}$. Specifically, since the number of states in OPEN[i+1] is less than $W_a|T|$, the time complexity for evaluating the states is $O(W_a|T|\cdot|P|C_r)$ in the worst case, and the time complexity of sorting states is $O(W_a|T|\log[W_a|T|])$.

The four parts are executed sequentially in each search step, and at most K search steps are performed in one iteration. Since $K \le uL_m$, $W_a \le W$, there is $O(K)*\{O([2W+1]\log[2W+1]) + O(W) + O(W_a|T|[|P|+|P|\cdot|T|]) + O(W_a|T|[u+W_a|T|\cdot|P|+W|P|]) + O(W_a|T|\cdot|P|C_r) + O(W_a|T|\log[W_a|T|])\} \le uL_m\{O([2W+1]\log[2W+1]) + O(W) + O(W|T|[|P|+|P|\cdot|T|]) + O(W|T|[u+W|T|\cdot|P|+W|P|]) + O(W|T|\cdot|P|C_r) + O(W|T|\log[W|T|])\} \approx O(uL_mW|T|[W|T|\cdot|P|+u+|P|C_r])$. Thus, the main search process of HHS has time complexity $O(uL_mW|T|[W|T|\cdot|P|+u+|P|C_r])$.

5. The four factor levels of each parameter are shown in Table I, and the RV values of HHS with different estimation functions are also recorded in Table I. Table II records the average RV (ARV) values of W_0 and δ under different factor levels.

Table I

Parameter levels of W_0 and δ, RV values of HHS under different parameter combinations

Factor level	Parameter W_0	Parameter δ	Experiment Number	Factor level W_0	Factor level δ	RV $f_1(M[\upsilon])$	RV $f_2(M[\upsilon])$	RV $f_3(M[\upsilon])$
1	10	1	1	1	1	241	226	234
2	20	2	2	1	2	230	233	230
3	30	3	3	1	3	242	223	231
4	40	4	4	1	4	227	224	224
			5	2	1	234	230	230
			6	2	2	235	231	230
			7	2	3	231	232	237
			8	2	4	235	236	231
			9	3	1	238	228	238
			10	3	2	236	241	237
			11	3	3	238	242	235
			12	3	4	238	243	230
			13	4	1	233	232	227
			14	4	2	237	232	227
			15	4	3	230	231	225
			16	4	4	233	232	233

Table II

ARV values of each parameter under different factor levels

Factor level	$f_1(M[\upsilon])$ W_0	$f_1(M[\upsilon])$ δ	$f_2(M[\upsilon])$ W_0	$f_2(M[\upsilon])$ δ	$f_3(M[\upsilon])$ W_0	$f_3(M[\upsilon])$ δ
1	235	236.5	**226.5**	**229**	229.75	232.25
2	233.75	234.5	232.25	234.25	232	231
3	237.5	235.25	238.5	232	235	232
4	**233.25**	**233.25**	231.75	233.75	**228**	**229.5**

6. The RPD value of HHS for each instance is recorded in Table III.

Table III

RPD values of HHS with and without DCP (%)

Instance	$f_1(M[\upsilon])$ Using DCP	$f_1(M[\upsilon])$ Without using DCP	$f_2(M[\upsilon])$ Using DCP	$f_2(M[\upsilon])$ Without using DCP	$f_3(M[\upsilon])$ Using DCP	$f_3(M[\upsilon])$ Without using DCP
In01	4.089	5.576	2.602	0.372	**0.000**	0.743
In02	7.337	3.804	**0.000**	1.359	1.902	1.359
In03	6.167	1.762	**0.000**	1.542	1.322	0.441
In04	10.261	5.037	**0.000**	2.799	1.866	5.224
In05	0.893	1.339	**0.000**	1.786	**0.000**	0.446
In06	2.614	5.556	4.902	4.248	**0.000**	1.634
In07	2.688	1.882	2.957	1.613	**0.000**	1.075

In08	**0.000**	3.401	2.268	3.855	0.454	0.680
In09	0.606	1.818	3.030	3.636	1.212	**0.000**
In10	4.018	2.679	3.571	2.232	**0.000**	2.232
In11	1.832	3.297	2.930	0.733	**0.000**	0.733
In12	**0.000**	0.309	0.309	1.543	0.617	**0.000**
In13	**0.000**	6.494	2.597	1.299	**0.000**	1.299
In14	2.778	6.019	1.389	2.315	0.463	**0.000**
In15	1.969	1.575	2.362	3.937	0.787	**0.000**
In16	4.319	4.319	**0.000**	0.332	0.332	2.326