Single Hoist Scheduling in No-Wait Flexible Flow Shop System

Wang Shilong1, Yao Wei1, and Zhou Jie1
1The State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing, China
Email: Email: {slwang@cqu.edu.cn, shuixinqiao@163.com, zj@cme.cqu.edu.cn}

Abstract—This paper studies the single hoist scheduling problem in no-wait flexible flow shop system with constant process times. The objective is to minimize the make span in the given job tasks. Different jobs have different process routes and process times. When a job starts, workstation must be free to process the job, and as soon as it is finished, the hoist must be free to move it to the next workstation. In this paper, a mathematics model is developed. A hybrid genetic algorithm and simulated annealing algorithm is proposed to find optimal solutions to the problem. Examples are given to demonstrate the effectiveness of the model in different types of problem.

Index Terms—Hoist Scheduling; No-wait Flexible Flow Shop System; Genetic Simulated Annealing Algorithm

I. INTRODUCTION

The paper addresses single hoist scheduling in a no-wait flexible flow shop system, in which material handling facilities such as hoists or cranes are widely used to handle material. In this system, the processing of each job has to be continuous, i.e. once is started on the first workstation, it must be processed according to processing routes without any interruption. Thus, when needed, the start of a job on the first workstation must be delayed in order to satisfy the no wait constraint. Every workstation has at least one machine. Some bottleneck workstations with long processing time may have several machines. Different kinds of products have different process routes and process time. The hoist or crane is in charge of transporting parts from one work station to another for the next operation. This manufacturing system is applied in different industries including chemical processing, food processing, metal and pharmaceutical production [1,2]. Such a flow shop scheduling problem (FSSP) is known as no-wait FSSP or Fm/nw/Cmax [3]. According to the research work by Garey and Johnson [4], the no-wait FSSP is NP-hard.

An ordinary flow shop model is a multi-stage production process, where the jobs have to visit all stages in the same order, whereas a flexible flow shop model is more realistic. Arthanari and Ramamurthy [5] first defined the flexible flow shop problem. A branch-and-bound method is proposed to get optimal solutions applied to very small instances. A detailed survey for the flexible flow shop problem has been given by Wang [6]. Metaheuristic algorithms have been developed rapidly in order to obtain a optical solution [7], such as simulated annealing algorithm (SA), tabu search algorithm (TS), genetic algorithm (GA), immune algorithm (IA). However, litter research has been done for flexible flow shop scheduling problem when material handle among workstations is taken into consideration. And in highly automatic manufacturing systems, material handle is especially important and often causes bottlenecks [8].

As far as hoist scheduling is concerned, many researchers have studied simple cycle schedules. Philips and Unger [9] proposed the first Integer Programming model (IP) for the single hoist cyclic scheduling problem. Shapiro and Nuttle [10] developed a branch-and-bound algorithm for the basic system and described the way to extend it to systems with multi-tank stages. 1-cyclic scheduling of a single hoist in extended electroplating lines with the time window is researched by Liu and Jiang [8], and multi-function tank and multi-tank constraints are taken into consideration. The model can find optical the start time and processing time in each workstation in cyclic production. Lei and Wang [11] presented a branch-and-bound algorithm for generating optical 2-cyclic scheduling with time window constraint. Ada Che, Chengbin Chu and Feng Chu [12] proposed a branch-and-bound algorithm for multi-cyclic hoist scheduling with constant processing times. A. Che and C. Chu [13] also proposed an analytical mathematical model and a branch-and bound algorithm for single track cyclic and multi-hoist scheduling problems. Previous approaches toward solving the cyclic hoist(s) scheduling have limitations that every kind of products have the same processing routes. When it comes to flexible flow shop manufacturing system, the algorithms should be improved. At the same time, they can’t optimize job tasks permutation.

This paper proposes a hybrid genetic algorithm and simulated annealing algorithm (GSA) for solving hoist scheduling in the no-wait flexible flow shop system to minimize makespan in an automatic electroplating line. Section 2 gives the description and formulation of this problem, in which process routes constraint, tank capacity constraint and hoist capacity constraint are taken into account. Section 3 translates the problem in to a permutation FSSP and proposes the detail procedures of GSSA. In section 4, simulation based on GA and GSA are presented. Finally, section 5 concludes the paper.

II. PROBLEM DESCRIPTION AND FORMULATION

The model considered in this paper is particularly relevant in an automatic electroplating line with a hoist for material handling. Such an electroplating line is composed of a sequence of chemical tanks. Each tank contains chemicals required for a special electroplating step in the processing of parts. In order balance the productivity of different tanks, some kinds of electroplating are sharing the same tanks, especially in electroplating pretreatment processes such as oil removal, acid clearing, water clearing, acid activating, and electroplating passivation.
processes. At the same time, more than one identical tank is used at same stages which need long processing times, which is called multi-tank stages. Figure 1 illustrates typical process routes in a basic system.

![Figure 1. Typical process routes in the electroplating line.](image)

Parts in one carrier are kept together as a unit throughout the entire system. In production, only one carrier can be introduced to the system at a time. When a carrier is introduced to the system, it is moved from the loading station to the tank for the first stage, and must be moved by the hoist and soaked in each chemical tank for a specific period of time according to process route sequentially. When all the process is finished, a carrier is moved to unloading station. In the real electroplate process, electroplate thickness is strongly related to the process time and current density. Process time and other parameters are always constant in an automatic electroplate product line for a certain carrier. So the parameters are always constant in an automatic electroplate product line for a certain carrier. The makespan in this problem is the max of finishing times in P, so the objective function to be minimized is

$$C_{max} = \text{Max} \left( F\left(P_i, m+1\right) \right) \quad (i \in P) \quad (1)$$

We make the following assumptions which correspond to what happens in reality [12]:

$$\theta_x \geq \delta_{x,y} \quad (x \in M) \quad (2)$$

$$\delta_{x,y} \leq \delta_{x,j} + \delta_{j,y} \quad (x, j, y \in M) \quad (3)$$

Relation (2) means that a move with carrier needs more time than a void move between the same pair of tanks, Relation (3) is widely known as the triangle inequality rule.

B. Process Routes Constraints

In this flexible flow shop system, variable V(Pi,x) is defined as follows to determine whether it is needed for Pi to be processed in Mi. While V(Pi,x) equal to zero, Pi will be processed in Mi, otherwise, Pi will be processed in Mi+1.

$$V(Pi,x) = \begin{cases} 0, & \text{if } T(i,x) = 0 \\ 1, & \text{if } T(i,x) = 0 \end{cases} \quad (i \in P, x \in M) \quad (4)$$

In the no-wait FSSP, as soon as S(Pi,0) is known, all starting times and finishing times of Pi in every stages are accordingly determined as follows:

$$S(Pi,x) = C(Pi,x) + \theta_{i+1} + R(Pi,x) \times V(Pi,x) \quad (i \in P, x \in M) \quad (5)$$

C. Tank Capacity Constraints

The tank capacity constraints require that when a carrier arrives at a stage, the stage must be free, which is defined as tank capacity constraints. A hoist can move only one carrier at a time. In order to satisfy no-wait constraint, the hoist must be free when any process is finished for any carrier in the system, which is defined as hoist capacity constraints.

A. Problems Parameters

To present the model for the basic problem, we define the following notation.

- Pi — Parts in carrier i.
- n — The number of carriers that keep parts needed to be processed.
- m — The number of process stages.
- P — Schedule or permutation of carriers to be processed. P = { P1, P2, ……, Pn}. P will be introduced to the line before Pi-1, i={2,3,……,n}.
- M — Process stages (process tanks in the line), M={ M1, M2,……, Mn}, M0 is loading workstation, and Mm+1 is unloading workstation.
- $T(Pi,x)$ — The process time of Pi in Mx, i={2,3,……,n}.
- $S(Pi,x)$ — The starting time of Pi in Mx.
- $F(Pi,x)$ — The finishing time of Pi in Mx.
- $R(Pi,x)$ — The preparing time of Pi in Mx.
- $\theta_x$ — The time required for the hoist to move a carrier from Mx to Mx+1.
- $\delta_{x,y}$ — The time required for the empty hoist traveling from Mx to My.
- TP(Pi,x) — The move task of Pi from Mx to its next stages.
- $C_{max}$ — The makespan of job tasks.

Here, $T(Pi,x), R(Pi,x)$, $\theta_x$ and $\delta_{x,y}$ are given. Pi will be introduced into the line at the time of $S(Pi, 0)$ and be finished all the process procedure at the time of $S(Pi, m+1)$. The preparing time of Pi contains the time the hoist load a carrier in the previous tank and then lay it to the next tank and time for other assistant work. Makespan in this problem is the max of finishing times in P, so the objective function to be minimized is

$$C_{max} = \text{Max} \left( F\left(P_i, m+1\right) \right) \quad (i \in P) \quad (1)$$

We make the following assumptions which correspond to what happens in reality [12]:

$$\theta_x \geq \delta_{x,y} \quad (x \in M) \quad (2)$$

$$\delta_{x,y} \leq \delta_{x,j} + \delta_{j,y} \quad (x, j, y \in M) \quad (3)$$

Relation (2) means that a move with carrier needs more time than a void move between the same pair of tanks, Relation (3) is widely known as the triangle inequality rule.

B. Process Routes Constraints

In this flexible flow shop system, variable V(Pi,x) is defined as follows to determine whether it is needed for Pi to be processed in Mi. While V(Pi,x) equal to zero, Pi will be processed in Mi, otherwise, Pi will be processed in Mi+1.

$$V(Pi,x) = \begin{cases} 0, & \text{if } T(i,x) = 0 \\ 1, & \text{if } T(i,x) = 0 \end{cases} \quad (i \in P, x \in M) \quad (4)$$

In the no-wait FSSP, as soon as S(Pi,0) is known, all starting times and finishing times of Pi in every stages are accordingly determined as follows:

$$S(Pi,x) = C(Pi,x) + \theta_{i+1} + R(Pi,x) \times V(Pi,x) \quad (i \in P, x \in M) \quad (5)$$

C. Tank Capacity Constraints

The tank capacity constraints require that when a carrier arrives at a stage, the stage must be free, which is to say, the previous carriers should have been finished and removed from the stage.

The stages in the electroplating line can be divided into two sets: single-tank set and multi-tank. We define set A as single-tank subset, where A={A1, A2,……, An}, and subset B as multi-tank set, where B={B1, B2,……, Bm}. It implies N+H=M.

In the simply flow shop line, Reference [14] gives relation (6) in order to satisfy the tank capacity constraint:

$$\left| C(Pi, x) - C(Pi, y) \right| \geq \left| C(Pi, x-1) - C(Pi, x-1) \right| + T(Pi, x) + T(Pi, y) \quad (i \in P, j \in P, i \neq j, x \in M) \quad (6)$$

As to single-tank subset A in the line of this paper, relation (6) can be written as follows. It means that when Pi arrives at Mi, Pi must be finished and removed from M and laid in the following stage.
\[
\begin{align*}
C(P_i, x) - C(P_{i+1}, x) &+ C(P_i, y) - C(P_{i+1}, y) \geq T(P_i, x) + \theta \\
+ R(P_i, x) \times V(P_i, x) + T(P_i, x) + \theta &+ R(P_{i+1}, x) \times V(P_{i+1}, x)
\end{align*}
\]
\[\text{if } i \in N; x \in A; y \in M \text{ and } y = x \pm 1 \]
\[\text{(7)}\]

As to subset B, it should at least one tank in Mx is free when Pi comes to Mx in set B. When Mx have h tanks, (h \geq 2), Pi is the carrier that has the earliest finishing time in Mx among h previous carriers that should be processed in Mx. So relation (8) must be satisfied.

\[
\begin{align*}
C(P_i, x) - C(P_{i+1}, x) &+ C(P_i, y) - C(P_{i+1}, y) \geq T(P_i, x) + \theta \\
+ R(P_i, x) \times V(P_i, x) + T(P_i, x) + \theta &+ R(P_{i+1}, x) \times V(P_{i+1}, x)
\end{align*}
\]
\[\text{if } i \in N; x \in B; y \in M \text{ and } y = x \pm 1 \]
\[\text{(8)}\]

D. Hoist capacity constraints

The hoist capacity constraints ensure that the hoist is scheduled to handle only one part at a time. If there is no conflict in the use of hoist between any pair of moves, it can be said that the hoist capacity constraints are satisfied. These constraints can be decomposed into three subsets:

- For any m \in M, there is no conflict in the use of hoist between any carrier in the line. When process routes constraints are taken into consideration, the hoist constraints between the moves among single-tanks become redundant. Therefore, we only need to consider the moves among the carriers in multi-tanks. Thus, we can obtain

\[
C(P_i, k) + \theta + \Delta_{k+1} \leq C(P_i, k)
\]
\[\text{(9)}\]

- For any i=(1,2,…,N-1), there is no conflict in the use of hoist between Pi and its successive carriers Pj. So move x of Pi should be done either before or after move j of move y of Pj. When the earlier move task is completed, there should be enough time for the void hoist moving to the tank where the carrier needed to be moved locates. Thus, we must have

\[
\begin{align*}
C(P_i, x) + \theta &+ R(P_i, x) \times V |(i,j)\rangle \leq C(P_j, y) \quad \text{if } C(P_i, x) \leq C(P_j, y) \\
C(P_j, y) + \theta &+ R(P_j, y) \times V |(i,j)\rangle \leq C(P_i, x) \quad \text{if } C(P_j, y) > C(P_i, x) \\
i = 1,2,\ldots,N-1, j = i+1,\ldots,N\rangle &\in \{1,2,\ldots,M-1\}; y = 2,3,\ldots,M-1;i = 1,2,\ldots,N-1 \}
\end{align*}
\]
\[\text{(10)}\]

- For any i=(1,2,…,N-1), there is no conflict in the use of hoist between Pi and its previous carriers Pj. Similarly, we can obtain

\[
\begin{align*}
C(P_i, x) + \theta &+ R(P_i, x) \times V |(i,j)\rangle \leq C(P_j, y) \quad \text{if } C(P_i, x) \leq C(P_j, y) \\
C(P_j, y) + \theta &+ R(P_j, y) \times V |(i,j)\rangle \leq C(P_i, x) \quad \text{if } C(P_j, y) > C(P_i, x) \\
i = 1,2,\ldots,N-1, j = i-1,\ldots,N\rangle &\in \{1,2,\ldots,M-1\}; y = 1,2,\ldots,M-1;i = 1,2,\ldots,N-1 \}
\end{align*}
\]
\[\text{(11)}\]

E. Makespan evaluation method

From the above analysis, the starting time of Pi at the loading workstation S(Pi, 0) should be computed so that S(Pi, x) and F(Pi, x) can be obtained. Then Cmax can be calculated when the job sequence is given. When S(Pi, x) is fixed, it must make sure there are no conflicts in tank capacity constraints and hoist capacity constraints when satisfying the process routes constraints.

Obviously, if Pi is started to be processed after the S(Pi, m+1), the above constraints will be satisfied. But it will make the makespan much longer. Let \(\Delta t\) be the minimum interval between S(Pi, 0) and S(Pi, m) which can satisfy the above constraints, a be the max T(Pi, x) (x \in A), and bi be the total processed time of Pi. So we can get that bi \(\leq \Delta t\) \(\leq b\). A variable step search algorithm can be applied to get \(\Delta t\).

\[
S(P_i, 0) = S(P_i, 0) + \Delta t
\]
\[\text{(12)}\]

\[
a_i = \text{Max}\{T(i,x)\} \quad (x \in A)
\]
\[\text{(13)}\]

\[
b_j = \sum_{i=1}^{M} (T(i,y) + \theta + R(P_j, y) \times V(i,y))
\]
\[\text{(14)}\]

III. THE CONSTRUCTION OF GSA FOR NO-WAIT FSSP

A GA approach is an iterative heuristic based on Darwin’s theory about “survival of the fittest and natural selection”. In order to use GA, a possible solution must be represented as a chromosome and a set of solutions is created to form a population. Problems are solved by an evolutionary process for population such as selection, crossover and mutation. This process continues until an improved solution cannot be obtained or until a prescribed number of populations have been generated.

A SA is an enhanced version of local search. Starting with an initially solution S, SA generates a neighbor S* of S, The objective values difference \(\Delta = F(S*) - F(S)\) is calculated. If \(\Delta < 0\), the neighbor S* is accepted. Otherwise, S* may be accepted with a possibility \(\exp(-\Delta /T)\), where T is simply a positive control parameter called temperature. The above process is known as Metropolis acceptance criterion. This temperature is periodically reduced every NT iterations, where NT denotes the epoch length, so that it moves gradually from a relatively high value to near zero as the algorithm processes according to a function referred to as the cooling schedule temperature. The above process is known as Metropolis acceptance criterion. This temperature is periodically reduced every NT iterations, where NT denotes the epoch length, so that it moves gradually from a relatively high value to near zero as the algorithm processes according to a function referred to as the cooling schedule.

The job permutation based on encoding scheme has been widely used for FSSP. Since the job permutation based representation bears the necessary information related to FSSP, it is applied in this paper. Each chromosome in the population is evaluated based on the makespan of the schedule it represents. The simple genetic algorithm (SGA) contains ordinary operators in coding, selection, crossover, and mutation, and it has disadvantages such as being premature, indifferent and lacking ability in local searches. And it is found that the performances of a SGA often depend on its parameter and
operator [15,16]. So a hybrid genetic algorithm based on GA and SA is proposed to solve the problem in this paper. A decision probability based on SA is employed to control the utilization of genetic mutation operation and the local research so as to prevent the premature convergence and concentrate computing effort on promising neighbor solutions. A hybrid GSA framework is illustrated in Figure 2, and the operators is summarized as follows:

1) Initialization: Generates an initial population with chromosomes that represent sequence of jobs. Initialize the parameters such as crossover possibility, mutation possibility, max run time and initial temperature.

2) Evaluation: Make span is used as criteria. And the fitness can be calculated as follows [17]:

\[
    \text{fitness} = \frac{a}{b + C_{\text{max}}} \quad (a > 0, b > 0)
\]  

(15)

3) Generation: Best individual is immediately included into the next generation. The rest are selected randomly from parent chromosomes in population based on fitness values. And then crossover and mutation is operated. The new individual is accepted according to Metropolis acceptance criterion, which allows for the probabilistic acceptance of higher-cost perturbed solutions as the next current solution and enables the algorithm to climb out of local optima. This operation continues until the new population is generated.

4) Cooling scheduling: Cooling scheduling is the parameter that controls the acceptance probability, and can make the algorithm give an approximate optical solution. Fixed scheduling that has decrement rules of the form \( t_k = t_0 \times a^k \) is applied, and the temperature decrement is kept proportionally constant with \( 0 < a < 1 \).

5) Termination Criteria: The GSA research is terminated when there is no improvement in fitness during a certain sequential steps or it reaches the number of generations.

IV. COMPUTATIONAL RESULT AND APPLICATION

A. Performance of GSA

To test the performance of the proposed GSA, numerical simulations are carried out. The initial temperature is the difference between makespan of the worst individual and makespan of the best individual. In this paper, roulette wheel selection is applied to select new individuals from old population. Partially mapping crossover (PMX) and SWAP are applied in this paper [15]. In order to compare the performance equally, problems of same dimensions have the same parameters. All programming is coded in C++ Builder 6.0 with an Intel Pentium 4 2.30 GHz CPU with 512MB of RAM. Each problem has run for ten times. \( C^* \) is the makespan of the best individual. \( \text{RE} \) is the relative error to \( C^* \), that is \( (C_{\text{NEH}} - C^*)/C^* \). \( \text{BRE} \) is the relative error of best individual to \( C^* \). \( \text{WRE} \) is the relative error of worst individual to \( C^* \). \( \text{AT} \) is the average run time. Results of Nawaz-Enscore-Ham (NEH) [18], GA and GSA trial runs are summarized in Table 1.

It is found that NEH is a good constructive algorithm for minimizing the objective function considered, though it cannot reach the best individual. Both GA and GSA can reach the best individual when the problem has small dimension. GA can even obtain a better individual in some problem dimension. When it comes to larger problem dimension, GSA is more effective. And average performance of GSA in all problem dimensions is always better than GA. Due to the termination criteria in this paper, \( \text{AT} \) in GSA is not much longer than GA. It means that GSA has a better performance and is more robust than GA in this problem.

![Figure 2. Framework of GSA](image-url)
This example involves 10 processing stages. There are 3 tanks in Stage 3, 3 tanks in Stage 7 and 2 tanks in Stage 9. Other stages have single tank. Process times of each carrier in each stage are illustrated in Table II. If the processing time is zero, it means that the carrier does not need to be processed in this stage. In order to simplify the problem, we suppose that assistance time of each carrier in each stage is 2 second, \( \theta \) is 2 second, and the empty move time between any two stages \( x \) and \( y \) is \( \delta_{x,y} = |x-y|+2 \), so that the triangle inequality rule is satisfied. The computation time used is 1.156 seconds.

The schedule is shown in Figure 3. The vertical and horizontal axes represent hoist location and time, respectively. Rectangles show the paths of products through the system. Solid rectangles are processing times while void ones are preparing times. The incline solid lines represent the loaded move of the hoist. The incline dotted lines are empty moves of the hoist. From this solution, we can find the scheme can satisfy process routes constraints, tank capacity constraints and hoist capacity constraints. Only two tanks are used in Stage 3 and Stage 7. The schedule permutation and the start time of each carrier in each stage are illustrated. Makespan is 1703 seconds. In fact, we can see that the hoist is always busy in the schedule, so it means that the number of tanks are closed to the limit that one hoist can dispose.

V. CONCLUSION

Material handling among workstations is critical and often leads to bottlenecks in automatic manufacturing systems such as automatic electroplating lines. Effective scheduling of hoist can achieve high productivity and is essential to product quality when processing times are critical. In this paper, hoist scheduling in no-wait flexible flow shop system with consistent processing time is studied. The problem can be transformed into job permutation scheduling that must satisfy process routes constraints, tank capacity constraints and hoist capacity constraints. Then a hybrid genetic algorithm and simulated annealing algorithm is proposed to find optical solution for scheduling in this system. Examples are given to demonstrate the performance of the model for different types of problems. Comparing to NEH and GA, the algorithm is robust and efficient in solving the problem. The solutions show that their constraints are very strict and necessary. The model can also be applied to optimize the design of flexible flow shop lines.

ACKNOWLEDGMENT

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
<th>M10</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>10</td>
<td>15</td>
<td>220</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>20</td>
</tr>
<tr>
<td>P2</td>
<td>20</td>
<td>20</td>
<td>220</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
<td>10</td>
</tr>
<tr>
<td>P3</td>
<td>25</td>
<td>30</td>
<td>250</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>160</td>
<td>15</td>
</tr>
<tr>
<td>P4</td>
<td>15</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>275</td>
<td>26</td>
<td>140</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>18</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>240</td>
<td>15</td>
<td>140</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>24</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>280</td>
<td>18</td>
<td>180</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>12</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>250</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>150</td>
<td>16</td>
</tr>
<tr>
<td>P8</td>
<td>24</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>300</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>130</td>
<td>15</td>
</tr>
<tr>
<td>P9</td>
<td>20</td>
<td>25</td>
<td>240</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>260</td>
<td>25</td>
<td>180</td>
<td>18</td>
</tr>
<tr>
<td>P10</td>
<td>30</td>
<td>30</td>
<td>270</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>200</td>
<td>20</td>
<td>140</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 3. Solutions for the example illustrated in Table 3.
This research was supported by National Key Project of Scientific and Technical Supporting Programs Funded by Ministry of Science & Technology of China During the 11th Five-year Plan (No. 2006BAF01A27).

REFERENCES


