IEMS Vol. 10, No. 1, pp. 74-83, March 2011.

Heuristic Approach for Lot Sizing and Scheduling Problem with State Dependent Setup Time

Junghee Han†
College of Business Administration
Kangwon National University, Chuncheon, Kangwon-Do 200-701, KOREA
Tel: +82-33-250-6151, E-mail: jhhan@kangwon.ac.kr

Received, April 1, 2010; Revised, June 28, 2010; Accepted, August 4, 2010

Abstract. In this paper, we consider a new lot-sizing and scheduling problem (LSSP) that minimizes the sum of production cost, setup cost and inventory cost. Setup carry-over, setup overlapping, state dependent setup time as well as demand splitting are considered. For this LSSP, we develop a mixed integer programming (MIP) model, of which the size does not increase even if we divide a time period into a number of micro time periods. Also, we develop an efficient heuristic algorithm by combining a decomposition scheme with a local search procedure. Test results show that the developed heuristic algorithm finds a good quality (in practice, even better) feasible solution using far less computation time compared with the CPLEX, a competitive MIP solver.

Keywords: Production Management, Scheduling, Lot-sizing and Scheduling, Heuristic

1. INTRODUCTION

In this paper, we consider a new kind of Lot-Sizing and Scheduling Problem (LSSP) permitting all of setup carry-over, setup overlapping and state dependent setup time. In some process industries producing, for example, steel (or metal alloy), operators seek to keep furnaces (or machines) warm for all time periods to avoid expensive shutdown/shutups (Toy and Berk, 2006; Berk *et al*., 2008) even though there are some time periods being in idle state for some furnaces. In idle state, furnaces are kept warm producing no output. However, if some furnaces are in idle state for sufficiently long time periods in successions, it may not be the best policy to keep furnaces warm for all time periods. If some furnaces are scheduled to be idle for a long time periods, we may consider shutdown of such furnaces in order to reduce fuel (or electric power) consumption for keeping them warm. The motivation of this study comes from the zinc alloy production lines at Seok-Po Refinery of Young Poong Co. operating 5 furnaces and producing 8 different zinc alloy items. Since the employees at this site are scheduled to work in three shifts for 8 hours each, setups can take place any time. Thus, it is realistic to consider setup carry-over and setup overlapping. Also, it is desirable to consider shutdowns and shutups since some furnaces at this site are occasionally reheated after the shutdown. Typically, reheating a furnace in shutdown state requires more time and cost than those in idle state. For example, reheating a furnace in shutdown state and setting it to be ready for a next item in Seok-Po Refinery takes about 10~15 hours depending on its capacity, while typical setup time in idle state takes only 1~2 hours. In this context, we consider a new LSSP permitting all of setup carry-over, setup overlapping and state dependent setup time, and call this problem by LSSP with State dependent Setup time (L_SS). In L_SS, we find an economic operation plan of machines considering setup cost in shutdown state and setup cost in idle state plus operation cost in idle state. Furthermore, we seek to find an economic lot-sizing and optimal assignment of items to machines differing in their capacities considering production cost and inventory cost.

Since this problem, L_SS, is a kind of LSSP, literature review is focused on LSSP. Typically, LSSP aims at minimizing the production cost and inventory cost by finding an optimal lot size per setup and an optimal production schedule simultaneously (Drexl and Hasse, 1995). LSSPs are classified into small bucket model and big bucket model depending on the maximum number of setups allowed on a machine per period (typically, denoting a day). Setups are allowed only once and more than once in small bucket and big bucket models, respectively. Drexl and Hasse (1995, 1996) addressed a small bucket model considering setup carry-over: proportional lot-sizing problem (PLSP). Setup carry-over enables us to skip duplicate setups when an item is produced on a machine over a number of time periods in

† : Corresponding Author

successions. Suerie (2006) extended the PLSP (Drexl and Hasse, 1996) by considering setup overlapping over two successive time periods. Setup may begin at a time period and may end at the next time period when setup overlapping is allowed. While, Sox and Gao (1999) and Gopalakrishnan (2000) considered a big bucket model allowing for setup carry-over but not setup overlapping on a single machine. Clark and Clark (2000) considered another big bucket model allowing for demand splitting, which enables us to produce an item at multiple machines simultaneously to meet the demand by the due. However, setup overlapping as well as setup carry-over are not considered. Recently, Meyr (2002) addressed a big bucket model considering both demand splitting and setup carry-over, while setup overlapping is not considered. Although there are numerous studies on big bucket model, for example, Fleischmann and Meyr (2000), Porkka *et al*. (2003) and Suerie and Stadtler (2003), to the best of my knowledge, setup overlapping is not considered anywhere in a big bucket framework. As a big bucket model, only Meyr (2002) and Han *et al*. (2007) considered demand splitting and setup carry-over. While, the L_SS addressed in this paper allows for multiple setups on a machine per period, demand splitting, setup carry-over, setup overlapping and state dependent setup cost/time. Thus, the L_SS can be conceptualized as a big bucket model extending Meyr (2002) to consider setup overlapping and state dependent setup cost/time additionally. Also, the L_SS can be conceptualized as an extension of Han *et al*. (2007) to consider state dependent setup time/cost.

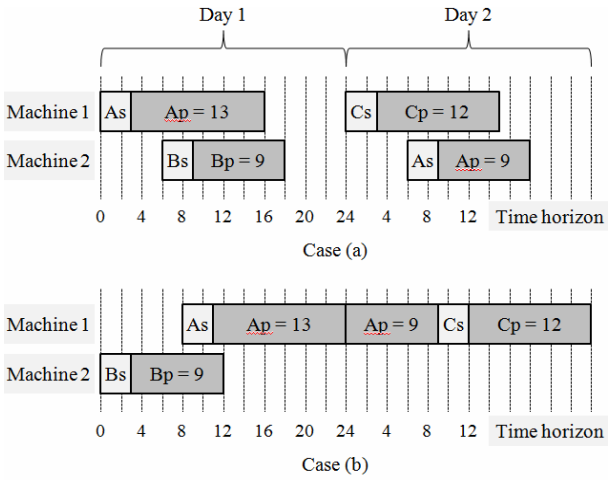

Figure 1. Illustration of the L_SS problem.

In Figure 1, we consider a simple example that illustrates the nature of the problem L_SS, where we consider two identical machines, three items A, B and C, time horizon spanning two days. Small letters 's' and 'p' next to A, B and C (denoting the item) indicate 'setup' and 'production', respectively. Thus, 'Ap = 13' on a row of 'Machine 1' denotes that "Item A is produced for 13 hours in a machine 1". Here, we assume that there exist two, one and one (production) dues for items A (Day 1 and Day 2), B (Day 1) and C (Day 2), respectively. Also, we assume that setup time of 3 hours is common for all items. To meet the demands of each item by its due, let us assume that we should produce item A for 13 hours during Day 1 and 9 hours during Day 2, item B for 9 hours during Day 1, and item C for 12 hours during Day 2, respectively. As a feasible solution, let us consider case (a), where we perform setups four times. While, we perform setups three times in case (b). Note that we produce item A on a machine 1 for 22 (= 13 + 9) hours in a row, which satisfies both Day 1 and Day2 dues of item A. Setup carry-over enables us to reduce setups for item A as shown in case (b). Also, note that in case (a) we may keep machine 2 warm for 12 hours from the 18th time period of Day 1 to the 6th time period of Day 2, while we may consider shutting down machine 2 from the 12th time period of Day 1 in case (b).

This paper is organized as follows. In Section 2, we develop a mixed integer programming (MIP) formulation for a generalized big bucket model considering demand splitting, setup carry-over, setup overlapping and state dependant setup cost/time. In Section 3, we propose a heuristic algorithm based on a decomposition scheme combined with a local search, and computational results are presented in Section 4. Section 5 concludes this paper.

2. FORMULATION

Let N be the set of items, and let K be the set of machines. Parameters and decision variables are defined below.

Parameters

- T : time horizon,
- L : maximum number of setups on a machine for T,
- $M(i)$: number of production requests of item $i \in N$ for T,
- T_{im} : due of the m-th production request of item $i \in N$, where $T_{i,1} < T_{i,2} < \cdots < T_{i,M(i)}$ for all $i \in N$,
- D_{im} : sum of demands of item $i \in N$ from T_{i1} to T_{im}, where $D_{i,1} < D_{i,2} < \cdots < D_{i,M(i)}$ for all $i \in N$,
- b_k : maximum production time per setup on a machine $k \in K$,
- s^{U}_{ki} : setup time for item $i \in N$ on a machine $k \in K$ in idle state,
- s^{D}_{ki} : setup time for item $i \in N$ on a machine $k \in K$ in shutdown state,
- p_{ki} : production rate of item $i \in N$ (per unit time) on a machine $k \in K$,
- α_{ki} : production cost of item $i \in N$ (per unit time) on a machine $k \in K$,
- β^{U}_{ki} : setup cost for item $i \in N$ on a machine $k \in K$ in idle state,
- β^{D}_{ki} : setup cost for item $i \in N$ on a machine $k \in K$ in shutdown state,

- γ_k : idle state operation cost on a machine $k \in K$ per unit time,
- π_{im} : inventory cost for holding item $i \in N$ of one unit from T_{im} to $T_{i,m+1}$.

Decision Variables

- x^{U}_{kli} : being equal to 1 if the l^{th} setup on a machine $k \in K$ in idle state is performed for item $i \in N$ (0 otherwise),
- x^{D}_{kli} : being equal to 1 if the l^{th} setup on a machine $k \in K$ in shutdown state is performed for item $i \in N$ (0 otherwise),
- y^{S}_{kl} : the time from which the l^{th} setup on a machine $k \in K$ begins,
- y^{E}_{kl} : the time when the l^{th} production on a machine $k \in K$ is completed,
- ξ^{U}_{kl} : time duration in idle state between the l^{th} and $(l-1)^{th}$ productions on a machine $k \in K$,
- ξ^{D}_{kl} : time duration in shutdown state between the l^{th} and $(l-1)^{th}$ productions on a machine $k \in K$,
- f_{kli} : time allocated to the l^{th} production (of item $i \in N$) on a machine $k \in K$,
- u^{0}_{klim} : being equal to 1 if the production of item $i \in N$ paired with the l^{th} setup on a machine $k \in K$ begins before T_{im} (0 otherwise),
- u^{1}_{klim} : being equal to 1 if the production of item $i \in N$ paired with the l^{th} setup on a machine $k \in K$ is completed before T_{im} (0 otherwise),
- w_{klim} : production time of item $i \in N$ paired with the l^{th} setup on a machine $k \in K$ contributing to meet D_{im},
- v_{im} : inventory of item $i \in N$ at T_{im}.

Using the notations defined above, we can formulate the problem L_SS as a mixed integer programming model (MIP) as follows.

L_SS: Minimize $\sum_{k \in K}\sum_{l \le L}\sum_{i \in N} \alpha_{ki} f_{kli} + \sum_{k \in K}\sum_{l \le L}\sum_{i \in N} (\beta^{U}_{ki} x^{U}_{kli} + \beta^{D}_{ki} x^{D}_{kli}) + \sum_{k \in K}\sum_{l \le L} \gamma_k \xi^{U}_{kl} + \sum_{i \in N}\sum_{m \le M(i)} \pi_{im} v_{im}$

Subject to

$$\sum_{i \in N} (x^{U}_{kli} + x^{D}_{kli}) \le 1 \qquad k \in K, l \le L, \quad (1)$$

$$\sum_{i \in N} (x^{U}_{kli} + x^{D}_{kli}) \ge \sum_{i \in N} (x^{U}_{k(l+1)i} + x^{D}_{k(l+1)i}) \qquad k \in K, l \le L-1, \quad (2)$$

$$\xi^{U}_{kl} + \xi^{D}_{kl} = y^{S}_{kl} - y^{E}_{k(l-1)} \qquad k \in K, l \le L, \quad (3)$$

$$\xi^{U}_{kl} \le T \sum_{i \in N} x^{U}_{kli} \qquad k \in K, l \le L, \quad (4)$$

$$\xi^{D}_{kl} \le T \sum_{i \in N} x^{D}_{kli} \qquad k \in K, l \le L, \quad (5)$$

$$y^{E}_{kl} = y^{S}_{kl} + \sum_{i \in N} (s^{U}_{ki} x^{U}_{kli} + s^{D}_{ki} x^{D}_{kli} + f_{kli}) \qquad k \in K, l \le L, \quad (6)$$

$$f_{kli} \le b_k (x^{U}_{kli} + x^{D}_{kli}) \qquad k \in K, l \le L, i \in N, \quad (7)$$

$$T_{im} u^{0}_{klim} - (T_{im} - s^{U}_{ki}) x^{U}_{kli} - (T_{im} - s^{D}_{ki}) x^{D}_{kli} + y^{S}_{kl} \ge 0 \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (8)$$

$$T u^{0}_{klim} - (T_{im} - s^{U}_{ki}) x^{U}_{kli} - (T_{im} - s^{D}_{ki}) x^{D}_{kli} + y^{S}_{kl} \le T \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (9)$$

$$T_{im} u^{1}_{klim} - T_{im} (x^{U}_{kli} + x^{D}_{kli}) + y^{E}_{kl} \ge 0 \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (10)$$

$$T u^{1}_{klim} - T_{im} (x^{U}_{kli} + x^{D}_{kli}) + y^{E}_{kl} \le T \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (11)$$

$$u^{0}_{klim}, u^{1}_{klim} \le x^{U}_{kli} + x^{D}_{kli} \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (12)$$

$$u^{1}_{klim} \le u^{0}_{klim} \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (13)$$

$$w_{klim} \le f_{kli} \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (14)$$

$$w_{klim} \le b_k u^{0}_{klim} \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (15)$$

$$w_{klim} - f_{kli} \ge b_k (u^{1}_{klim} - 1) \qquad k \in K, l \le L, i \in N, m \le M(i), \quad (16)$$

$$y^{S}_{kl} + s^{U}_{ki} x^{U}_{kli} + s^{D}_{ki} x^{D}_{kli} + w_{klim} - T_{im} \le T(1 - u^{0}_{klim} + u^{1}_{klim}) \quad k \in K, l \le L, i \in N, m \le M(i), \quad (17)$$

$$y^{S}_{kl} + s^{U}_{ki} x^{U}_{kli} + s^{D}_{ki} x^{D}_{kli} + w_{klim} - T_{im} \ge -T(1 - u^{0}_{klim} + u^{1}_{klim}) \quad k \in K, l \le L, i \in N, m \le M(i), \quad (18)$$

$$\sum_{k \in K}\sum_{l \le L} p_{ki} w_{klim} = D_{im} + v_{im} \quad i \in N, m \le M(i), \quad (19)$$

$$x^{U}_{kli}, x^{D}_{kli} \in \{0, 1\} \qquad k \in K, l \le L, i \in N,$$

$$u^{0}_{klim}, u^{1}_{klim} \in \{0, 1\} \qquad k \in K, l \le L, i \in N, m \le M(i),$$

all other variables are non-negative and continuous, where y^{E}_{k0} indicates the earliest time we can start any job at machine $k \in K$.

Remark 1: In this paper, we set $y^{E}_{k0} = 0$ for all $k \in K$. Also, we set $x^{D}_{k1i} = 0$ for all $k \in K$ and $i \in N$. That is, we assume idle state at y^{E}_{k0} for all $k \in K$.

The objective function minimizes the sum of production cost, setup cost, operating cost in idle state, and inventory cost that are expressed by the first, second, third and fourth terms, respectively. Constraint (1) ensures that at most one item can be setup at each production sequence on a machine, and that setups in idle state and in shutdown state should be distinguished from each other. Constraint (2) removes symmetry of setup sequences on a machine. Constraint (3) expresses that setup for a next item may begin when the production in progress is completed. Constraints (4) and (5) express the time duration in idle (or shutdown) state between any pair of consecutive production runs. Constraint (6) prohibits idle (and shutdown) state between setup and production. However, we allow idle (and shutdown) state between production and setup for a next item. Constraint (7) limits maximum production time on a machine per setup. Constraints (8)~(13) determine the values of u variables by the values of x and y variables. Constraints (14)~(18) determine the values of w variables by the values of f, u, x and y variables. Below we explain constraints (8)~(18) in detail. Constrain (19) expresses that backlogging is not allowed. That is, all the demands should be satisfied by their dues.

Let us consider constraints (8)~(18) calculating the exact values of w variables. For the simplicity of explanation, we assume that $x^{D}_{kli} = 0$. Now, let us assume that item $i \in N$ is assigned to the l^{th} production sequence on a machine $k \in K$ ($x^{U}_{kli} = 1$) since otherwise ($x^{U}_{kli} = 0$), it is clear that $f_{kli} = 0$ from constraint (7), which in turn forces that $w_{klim} = 0$ for all $m \le M(i)$ from constraints (14). For some $m \le M(i)$, we may consider five cases (a) ~(e) depending on the values of T_{im}, y^{S}_{kl} and f_{kli}. First, we consider case (a), where $w_{klim} = f_{kli}$ by constraints (14) and (16). For case (b), we see that $w_{klim} = 0$ from con-

straint (15). For case (*c*), we see that $w_{klim} = T_{im} - (y^S_{kl} + s^U_{ki})$ from constraints (17) and (18). As a special case of (*a*), we consider case (*d*), where u^1_{klim} can take either 1 or 0 from constraints (10) and (11). If $u^1_{klim} = 1$, we see that $w_{klim} = f_{kli}$ from constraints (14) and (16). While, if $u^1_{klim} = 0$, we see that $w_{klim} = T_{im} - (y^S_{kl} + s^U_{ki})$ from constraints (17) and (18), which is equal to f_{kli}. Also, as a special case of (*c*), we consider case (*e*), where u^0_{klim} can take either 0 or 1 from constraints (8) and (9). If $u^0_{klim} = 0$, it is clear that $w_{klim} = 0$ from constraint (15). While, if $u^0_{klim} = 1$, we see that $w_{klim} = T_{im} - (y^S_{kl} + s^U_{ki})$ from constraints (17) and (18), which is also equal to 0.

Now let us investigate some interesting features inherent in the model L_SS. Instead of using the indices denoting the time periodes, for example indices *t*'s, we define decision variables by production (or setup) sequence with index $l \le L$ for each machine. The advantage of this expression is that the size of MIP formulation for L_SS does not increase even if we increase the time horizon *T* or even if we increase the precision of a time period. Using the following example, let us explain this in detail.

Example 1: For two items *A* and *B*, assume that each item has two production requests: $T_{A,1} = 3t$, $T_{A,2} = 10t$, $T_{B,1} = 5.5t$ and $T_{B,2} = 7t$, where *t* indicates the unit time period (for example, a day). Then, time horizon *T* becomes 10*t*. Suppose that we express the problem using decision variables employing index *t* denoting time period. Due to $T_{B,1} = 5.5t$, which is not a integer multiple of *t*, we should meet $D_{B,1}$ until 5*t* not 5.5*t*, or we should divide a time period *t* into two micro time periods, in which case the number of time periods in the time horizon *T* doubles.

Slicing a time period into a number of micro time periods may be viable in theory, but this increases the formulation size. If the length of a (micro) time period *t* is defined sufficiently small, and if we can handle the formulation that may be extremly big in size, we may obtain highly elaborate production schedule in view of time utilization. However, in paractice, the problem may become computaionally intractable due to huge formulation size. In recognition of this problem, Meyr (2000, 2002) developed a formulation, where setup overlapping is not considered, having the flexibility to define different length of a micro time period for each (macro) time period, which can partially circumvent this problem. This approach enables us to increase the precision at some (macro) time periods selectively. Another approach is to use index *l* indicating the setup sequence for each machine and for each time period *t* proposed by Clark and Clark (2000). They use both indices *t*'s and *l*'s to express time periods and setup sequences on a machine, respectively. However, they consider neither setup overlapping nor setup carry-over due to the complexity of handling different index types.

Unlike the big bucket models considered so far, the size of formulation L_SS is not affected by the precision of a (micro or macro) time period since index *t* indicating the time period is not used at all. We consider time period as a continuum not discrete time interval.

Optimally solving medium or large size big bucket LSSP problems seems quite chanllenging when all of demand splitting, setup carry-over and setup overla-pping are considered. Note that the maximum problem size considered in the literture is quite small. For ex-ample, $|N| = 6$, $|K| = 2$, $T = 5$ (Clark and Clark, 2000), $|N| = 19$, $|K| = 2$, $T = 8$ (Meyr, 2002), $|N| = 8$, $|K| = 1$, $T = 8$ (Sox and Gao, 1999).

3. HEURISTICS

Gupta and Magnusson (2005) and Porkka *et al*. (2003) developed local search algorithms for a single machine LSSP considering setup carry-over, and Sox and Gao (1999) developed a Lagrangean-relaxation heuristic algorithm for the same LSSP. Suerie and Stadtler (2003) developed a heuristic algorithm for a multiple machine LSSP allowing for setup carry-over. Although there exist a number of heuristic algorithms on LSSP, for example, Drexl and Haase (1996) and Fleischmann and Meyr (1997), to the best of my knowledge, all of setup carry-over, setup overlapping and state dependent setup cost/time are not considered anywhere. Below, we develop a simple but effective heuristic algorithm by devising a decomposition scheme coupled with a local search procedure. First, we present an initial procedure to find a feasible solution. Then, we develop an improving procedure that solves two sub-problems obtained from L_SS in iterations for a given time limit.

3.1 Initial Procedure

Define t_k as the time from which machine $k \in K$ becomes available, and define Q_{im} as the cumulative quantity of item *i* produced until T_{im}. Using the notations above, the initial procedure is described as follows.

Step 0: Set $t_k = 0$ for $k \in K$ and $Q_{im} = 0$ for $i \in N$ and $m \le M(i)$. Define a list of (i, m) pairs, and denote it by $PL = \{(i, m): i \in N, m \le M(i)\}$.

Step 1: Sort (i, m)'s in *PL* in increasing order of T_{im}.

Step 2: If $D_{im} \le Q_{im}$ for all (i, m)'s in *PL*, exit. Otherwise, pick a pair (i, m) with the lowest index in *PL* such that $D_{im} > Q_{im}$, and go to Step 3.

Step 3: Define $K(i, m) = \{k \in K: t_k + s^U_{ki} + (D_{im} - Q_{im})/p_{ki} \le T_{im}\}$. If $K(i, m) \neq \varnothing$, assign (i, m) to a machine $k \in K(i, m)$ requiring minimum cost to produce $\min\{D_{im} - Q_{im}, b_k \times p_{ki}\}$. While, if $K(i, m) = \varnothing$, assign to a machine $k \in K$ that maximizes the production of item *i* until T_{im}, $\min\{(T_{im} - t_k - s^U_{ki}), b_k\} \times p_{ki}$. Then, update Q_{im} and t_k, and go to Step 2.

Note that we consider idle state setup time s^U_{ki} to determine the machine to produce item i in Step 3. That is, $x^D_{kli} = 0$ for $k \in K$, $l \le L$ and $i \in N$ in an initial feasible solution. Also, when updating t_k in Step 3, we set t_k to the completion time of current production on machine k. That is, $\xi^U_{kl} + \xi^D_{kl} = 0$ for $k \in K$ and $l \le L$ in an initial feasible solution. By eliminating idle time for two consecutive production tasks for each machine, inventory cost may increase. However, we can increase the chance to find a feasible solution.

3.2 Improving Procedure

In this subsection, we develop two heuristic procedures H1 and H2, where H2 is a modification of H1.

Heuristic 1 (H1)

For a given solution S, define $X(S) = \{(k, l, i): x^U_{kli} = 1$ for $k \in K$, $l \le L$ and $i \in N\}$, and define $Z(S)$ as the objective value of S. Given an initial solution S_0, we attempt to reduce the total cost by choosing $x^U_{kli} = 1$ or $x^D_{kli} = 1$ for $(k, l, i) \in X(S_0)$ and by adjusting the values of y^S and u variables. For this purpose, we optimally solve L_SS($x^U + x^D, f$) and obtain an updates solution S_1, where L_SS(x^U+x^D, f) denotes the L_SS such that

1) constraint (1) for $(k, l, i) \in X(S_0)$ is replaced by $x^U_{kli} + x^D_{kli} = 1$ and constraint (1) for $(k, l, i) \notin X(S_0)$ is replaced by $x^U_{kli} + x^D_{kli} = 0$, and that
2) f variables are fixed based on S_0.

Inventory cost of S_1 would be smaller than that of S_0. Then, we attempt to further reduce the total cost by optimizing the values of f and y variables after fixing x^U, x^D and u variables according to S_1. For this purpose, we optimally solve L_SS(x^U, x^D, u) and obtain a further updated solution S_2, where L_SS(x^U, x^D, u) denotes the L_SS such that x^U, x^D and u are fixed according to S_1. Note that L_SS(x^U, x^D, u) is a linear programming (LP) problem since all the binary variables x and u are fixed. Based on S_2, we again generate a new PL, from which we generate a new S_0 again. This process is repeated for a given time limit. Detailed procedure for generating a new PL based on S_2 is explained along with the overall procedure described below.

Step 0 (Initialize). Define an elite priority list EPL as a collection of PL's, and set $EPL = \varnothing$.
Step 1 (Find an initial solution). Obtain a feasible solution S_0 by running the initial procedure. Recall that $x^D_{kli} = 0$ for all $k \in K$, $l \le L$ and $i \in N$ in an initial solution S_0.
Step 2 (Reducing total cost).
Step 2.1 Solve L_SS($x^U + x^D, f$), where $x^U + x^D$ and f are fixed according to S_0, and obtain an updated solution S_1.
Step 2.2 Solve a linear programming (LP) problem L_SS (x^U, x^D, u), where x^U, x^D and u are fixed according to S_1, and obtain an updated solution S_2.
Step 3 (Update EPL). Add the PL associated with the current solution S_2 to EPL. If $|EPL| > MaxEPL$, discard one PL from EPL having the largest objective value.
Step 4 (Find an alternative solution). Define $cost(i, m)$ as the sum of setup cost and production cost in S_2 to satisfy D_{im} divided by D_{im} for $i \in N$ and $m \le M(i)$.
Step 4.0 Set $cnt = 0$.
Step 4.1 Pick an arbitrary (i, m) in the PL associated with the current solution S_2 satisfying that $cost(i, m) \ge wgt \times \max\{cost(i, m): i \in N$ and $m \le M(i)\}$, where wgt is a fixed parameter between 0 and 1. Increase the priority of such a pair (i, m) in PL by one, and let $cnt = cnt + 1$.
Step 4.2 If the resulting PL is found in EPL, go to Step 4.3, else, go to Step 4.4.
Step 4.3 If $cnt \ge MaxCnt$, go to Step 5, else, go to Step 4.1.
Step 4.4 Find a feasible solution S_0 using the initial procedure based on the updated PL, and go to Step 2. If feasible solution S_0 is not found, go to Step 4.1.
Step 5 (Restart). Pick an arbitrary PL from EPL, and go to Step 4.

From the preliminary test, we observed that an optimal solution is mainly different from near optimal solutions in the configuration of setups. For example, to satisfy D_{im} for some (i, m), an optimal solution uses only one machine, while near optimal solutions usually split D_{im} to multiple demand segments and assigns them to multiple machines, in which case setup cost increases. From this observation, we defined $cost(i, m)$ as the sum of setup cost and production cost in S_2 to satisfy D_{im} divided by D_{im} for $i \in N$ and $m \le M(i)$. Also, we increase the priority of a selected pair (i, m) in Step 4.

Heuristic 2 (H2)

The second heuristic procedure H2 differs from H1 only in Step 2.1. In Step 2.1 of H1, we deal with binary variables x^U_{kli}, x^D_{kli} and u simultaneously. Although computing time to solve L_SS(x^U+x^D, f) optimally is far smaller than that to optimally solve L_SS, we consider an alternative approach to find a good quality solution to L_SS(x^U+x^D, f). For this purpose, we devise a two-phase procedure for Step 2.1 of H1. In the first phase, we set $x^U_{kli} = 1$ for $(k, l, i) \in X(S_0)$ and $x^U_{kli} = x^D_{kli} = 0$ for $(k, l, i) \notin X(S_0)$. Denote the L_SS such that x^U, x^D and f are fixed according to the solution S_0 by L_SS(x^U, x^D, f). By solving L_SS(x^U, x^D, f), we can fix u variables along with other continuous variables, and can obtain an updated solution S_1. In the second phase, given fixed values of u variables, we examine if changing from $x^U_{kli} = 1$ to $x^D_{kli} = 1$ for each $(k, l, i) \in X(S_1)$ reduces the total cost. For this purpose, we consider L_SS($x^U + x^D, u$) denoting the L_SS such that

1) constraint (1) for $(k, l, i) \in X$ is replaced with $x^U_{kli} + x^D_{kli} = 1$ and constraint (1) for $(k, l, i) \notin X(S_1)$ is re-

placed with $x^U_{kli} + x^D_{kli} = 0$, and that

2) u variables are fixed according to S_1.

That is, given fixed values of u variables, we choose one between $x^U_{kli} = 1$ and $x^D_{kli} = 1$ for all $(k, l, i) \in X(S_1)$ by solving L_SS($x^U + x^D$, u), which produces a further updated solution S_2. The problem L_SS($x^U + x^D$, u) can be conceptualized as a bi-partition problem over $X(S_1)$. The overall procedure of H2 is as follows.

Step 0 (Initialize). Define an elite priority list *EPL* as a collection of *PL's*, and set $EPL = \varnothing$.

Step 1 (Find an initial solution). Find a feasible solution S_0 by running the initial procedure. Recall that $x^D_{kli} = 0$ for all $k \in K$, $l \le L$ and $i \in N$ in an initial solution S_0.

Step 2 (Reducing total cost).

Step 2.1 Solve L_SS(x^U, x^D, f), where x^U, x^D and f are fixed according to S_0, and obtain an updated solution S_1.

Step 2.2 Examine if changing $x^U_{kli} = 1$ to $x^D_{kli} = 1$ for $(k, l, i) \in X(S_1)$ reduces the total cost by solving L_SS($x^U + x^D$, u), and obtain an updated solution S_2.

Step 2.3 Solve a linear programming (LP) problem L_SS (x^U, x^D, u), where x^U, x^D and u are fixed according to S_2, and obtain an updated solution S_3.

Step 3 (Update *EPL*). Add the *PL* associated with the current solution S_3 to *EPL*. If $|EPL| > MaxEPL$, discard one *PL* from *EPL* having the largest objective value.

Step 4 (Find an alternative solution). Define $cost(i, m)$ as the sum of setup cost and production cost in S_3 to satisfy D_{im} divided by D_{im} for $i \in N$ and $m \le M(i)$.

Step 4.0 Set $cnt = 0$.

Step 4.1 Pick an arbitrary (i, m) in the *PL* associated with the current solution S_3 satisfying that $cost(i, m) \ge wgt \times \max\{cost\ (i, m): i \in N \text{ and } m \le M(i)\}$, where *wgt* is a fixed parameter between 0 and 1. Increase the priority of such a pair (i, m) in *PL* by one, and let $cnt = cnt + 1$.

Step 4.2 If the resulting *PL* is found in *EPL*, go to Step 4.3, else, go to Step 4.4.

Step 4.3 If $cnt \ge MaxCnt$, go to Step 5, else, go to Step 4.1.

Step 4.4 Find a feasible solution S_0 using the initial procedure based on the updated *PL*, and go to Step 2. If feasible solution S_0 is not found, go to Step 4.1.

Step 5 (Restart with a *PL*). Pick an arbitrary *PL* from *EPL*, and go to Step 4.

4. COMPUTATIONAL RESULTS

We have generated test problems as follows.

- Set pseudo capability of each machine: $capa_k = 1 + U[1, 2]$, where U[] denotes uniform distribution.
- Setup time/cost: $s^U_{ki} = \lfloor 2 \times capa_k + U[3, 5] \rfloor$, $s^D_{ki} = 3 \times s^U_{ki}$, $\beta^U_{ki} = s^U_{ki} \times 500$ and $\beta^D_{ki} = 3 \times \beta^U_{ki}$.
- Production rate/cost: $p_{ki} = \lfloor 3 \times capa_k \rfloor$ and $\alpha_{ki} = p_{ki} \times 10$,
- Idle state operation cost: $\gamma_k = \lfloor 2 \times capa_k \rfloor$,
- Maximum production time per setup: $b_k = \lfloor U[5, 7] \rfloor$,
- Number of production requests: $M(i) = \lfloor U[T/2, T+1] \rfloor$,
- Time dues of each production request:
  - $T_{i,1} = 2 \times min\{s^U_{ki} + b_k: i \in N, k \in K\} + \lfloor U[0, 1] \times T/max\{M(i): i \in N\} \rfloor$ for $i \in N$,
  - $T_{im} = T_{i,m-1} + min\{s^U_{ki} + b_k: i \in N, k \in K\} + 2 \times \lfloor U[0, 1] \times T/max\{M(i): i \in N\} \rfloor$ for $i \in N$ and $m = 2, \cdots, M(i)$,
- Cumulative demands: $D_{im} = D_{i,m-1} + U[10, 20]$, where $D_{i0} = 0$, for $i \in N$ and $m \le M(i)$.
- Inventory costs: $\pi_{im} = 5 \times (T_{i,m+1} - T_{im})$ for $m = 1, \cdots, M(i)-1$, and $\pi_{i,M(i)} = 100$.

There are three parameters used in our heuristic algorithms; *MaxEPL*, *MaxCnt* and *wgt*. We set $MaxEPL = |N|$, $MaxCnt = \lfloor |N|/2 \rfloor$ and $wgt = 0.8$. Preliminary test results show that the best solution is slowly updated as *MaxEPL* and *MaxCnt* is too large. This implies that best solution is likely to be found from a good *PL*, which accords with our strategy that restarting from an elite *PL*. Also, we observed that it is better to set *MaxCnt* smaller than *MaxEPL* in terms of computing time and solution quality. As we decrease *wgt*, randomness in choosing the (i, m) to change its priority in the *PL* increases. To our experience, random reordering of the current *PL*, for example, setting $wgt = 0$, provides poor solution quality. However, in order to give some diversity of reordering the *PL*, we set $wgt = 0.8$.

Heuristic algorithms are coded in *C* using callable libraries of CPLEX Version 9.0. All test runs were executed on a Pentium IV (CPU: 2.8GHz, RAM: 2GB). Computational results are displayed in Tables 1~6. LP-relaxation bound of 'L_SS' denotes the optimal objective value to the LP-relaxation of L_SS at the root node of branch-and-bound tree. We interrupted the CPLEX optimization procedure at 10,000 seconds if it is not terminated until then. L_SS upper bounds associated with the mark '10K' in the column 'Elapsed Time' may not be the optimal objective values. In order to evaluate the quality of heuristic upper bounds in comparison with L_SS upper bounds obtained by CPLEX optimization procedure, we present 'Ratio', where

Ratio = (H1 or H2 upper bound-L_SS upper bound) / L_SS upper bound × 100%.

We have run the heuristic algorithms for 1,000 seconds. Note that the 'Ratio' of the heuristic algorithms H1 and H2 does not exceed 1.8%(#6) and 2.8%(#6), respectively, for the problems in Table 1. While, the computing time for H1 and H2 is far smaller than that for CPLEX based on the formulation L_SS. Also, the H1 outperforms the H2 in terms of solution quality. In

Table 2~Table 4, we report computational results for larger L and T. For the test problems in Table 2~Table 4, it is not guaranteed that the H1 always finds a feasible solution. For these test problems, we denote them by '-' marks. While, note that the H2 found a feasible solution for all test problems in Table 2~Table 4 although the solution quality of H2 is worse than that H1 finds. Also, note that for all test problems in Table 2~Table 4, CPLEX optimization procedure failed to find an optimal solution within 10,000 seconds, while the heuristic

Table 1. Comparison of upper bounds ($|N| = 5$, $|K| = 3$, $L = 5$, $T = 5$ days).

No	LP bound	Upper bound			Ratio		Elapsed Time		
		L_SS	H1	H2	H1	H2	L_SS	H1	H2
1	332.01	426.47	423.87	438.07	-0.6	0.4	10K	1K	1K
2	296.60	421.50	412.07	422.63	-2.2	0.3	10K	1K	1K
3	359.11	470.24	463.10	466.10	-1.5	-0.9	10K	1K	1K
4	307.61	422.43	422.09	422.09	-0.9	-0.9	10K	1K	1K
5	300.19	399.67	404.67	408.47	1.3	2.2	6740	1K	1K
6	312.61	408.84	416.0	420.50	1.8	2.8	3690	1K	1K
7	337.08	425.34	424.13	434.40	-0.3	2.1	10K	1K	1K
8	312.62	420.06	425.23	428.23	1.2	1.9	9142	1K	1K
9	300.50	388.70	390.10	396.47	0.4	2.0	5412	1K	1K
10	355.63	460.89	463.84	470.22	0.6	2.0	10K	1K	1K
11	329.55	418.04	419.10	428.67	0.3	2.5	9881	1K	1K
12	327.46	411.54	409.20	421.07	-0.6	2.3	10K	1K	1K
13	312.62	420.57	412.90	427.90	-1.8	1.7	10K	1K	1K
14	317.71	444.75	444.26	451.49	-0.1	1.5	10K	1K	1K
15	325.22	442.18	445.46	453.19	0.7	2.5	10K	1K	1K
16	305.39	405.40	405.10	413.60	-0.1	2.0	10K	1K	1K
17	320.92	422.40	427.27	428.33	1.2	1.4	10K	1K	1K
18	315.0	400.50	400.50	402.60	0	0.5	6204	1K	1K
19	387.09	503.97	501.97	511.23	-0.4	1.4	10K	1K	1K
20	289.86	377.84	378.67	381.77	0.8	1.0	5032	1K	1K

Table 2. Comparison of upper bounds ($|N| = 5$, $|K| = 3$, $L = 10$, $T = 10$ days).

No	LP bound	Upper bound			Ratio		Elapsed Time		
		L_SS	H1	H2	H1	H2	L_SS	H1	H2
1	805.62	1384.58	1164.38	1241.66	-15.9	-10.3	10K	1K	1K
2	933.40	1381.67	-	1268.43	-	-8.2	10K	1K	1K
3	758.62	1300.08	1178.43	1251.83	-9.4	-3.7	10K	1K	1K
4	872.13	1296.06	1287.28	1308.97	-0.6	1.0	10K	1K	1K
5	734.32	1237.41	1121.92	1182.75	-9.3	-4.4	10K	1K	1K
6	805.80	1112.35	-	1104.60	-	-0.7	10K	1K	1K
7	897.92	1355.58	1274.27	1370.50	-6.0	1.1	10K	1K	1K
8	858.69	1256.55	1216.23	1262.18	-3.2	0.4	10K	1K	1K
9	810.02	1176.36	-	1131.13	-	-3.8	10K	1K	1K
10	746.73	1191.01	1186.23	1206.37	-0.4	1.3	10K	1K	1K
11	811.99	1242.72	-	1245.52	-	0.2	10K	1K	1K
12	845.36	1261.03	-	1217.03	-	-3.5	10K	1K	1K
13	880.67	1512.04	1404.96	1421.23	-7.1	-6.0	10K	1K	1K
14	852.76	1224.58	1172.25	1177.38	-4.2	-3.9	10K	1K	1K
15	912.53	1366.32	1337.23	1363.83	-2.1	-0.2	10K	1K	1K
16	966.82	1276.70	1232.28	1261.43	-3.4	-1.2	10K	1K	1K
17	960.53	1591.58	1537.22	1573.28	-3.4	-1.1	10K	1K	1K
18	840.16	1243.05	1240.32	1267.15	-0.2	1.9	10K	1K	1K
19	832.14	1326.38	-	1294.88	-	-2.3	10K	1K	1K
20	706.97	1290.52	1183.33	1208.76	-8.3	-6.3	10K	1K	1K

algorithm H2 provides a better or equivalently good feasible solution for 16 test problems out of 20 in Table 2, 17 test problems out of 20 in Table 3, and 20 test problems out of 20 in Table 4, respectively.

In Table 5 and Table 6, we report additional test results for larger $|N| = 10$ and $|K| = 6$, which is larger than real-world problems at Seok-Po Refinery of Young Poong Co. producing 8 different zinc alloy items with 5 furnaces ($|N| = 8$ and $|K| = 5$). Note that the H2 still provides a feasible solution of good quality (see 'Ratio' of

Table 3. Comparison of upper bounds ($|N| = 5$, $|K| = 3$, $L = 20$, $T = 20$ days).

No	LP bound	Upper bound			Ratio		Elapsed Time		
		L_SS	H1	H2	H1	H2	L_SS	H1	H2
1	1001.55	1521.75	1507.33	1522.02	-0.9	0.1	10K	1K	1K
2	832.88	1203.00	-	1169.80	-	-2.8	10K	1K	1K
3	936.19	1380.79	1331.58	1331.58	-3.6	-3.6	10K	1K	1K
4	1733.86	3247.98	-	2964.82	-	-8.7	10K	1K	1K
5	1809.94	2438.30	-	2409.67	-	-1.2	10K	1K	1K
6	987.50	1318.44	1275.8	1275.80	-3.2	-3.2	10K	1K	1K
7	967.78	1393.54	-	1306.80	-	-6.2	10K	1K	1K
8	900.53	1266.38	-	1210.67	-	-4.4	10K	1K	1K
9	1064.88	1537.14	-	1484.20	-	-3.4	10K	1K	1K
10	1024.73	1437.87	1420.03	1448.60	-1.2	0.7	10K	1K	1K
11	1095.78	1492.70	1461.34	1470.33	-2.1	-1.5	10K	1K	1K
12	925.42	1593.88	-	1537.35	-	-3.6	10K	1K	1K
13	973.41	1428.68	1323.45	1369.53	-7.3	-4.1	10K	1K	1K
14	1161.64	1576.33	1423.65	1489.30	-9.7	-5.5	10K	1K	1K
15	1043.26	1553.66	-	1491.30	-	-4.0	10K	1K	1K
16	933.15	1505.96	1475.91	1508.60	-2.0	0.2	10K	1K	1K
17	1006.64	1469.63	1402.33	1433.24	-4.6	-2.5	10K	1K	1K
18	954.64	1364.55	1322.58	1342.20	-3.1	-1.6	10K	1K	1K
19	910.26	1395.98	-	1322.78	-	-5.2	10K	1K	1K
20	921.44	1410.10	1312.43	1345.80	-6.9	-4.6	10K	1K	1K

Table 4. Comparison of upper bounds ($|N| = 5$, $|K| = 3$, $L = 30$, $T = 30$ days).

No	LP bound	Upper bound			Ratio		Elapsed Time		
		L_SS	H1	H2	H1	H2	L_SS	H1	H2
1	1395.45	3358.27	-	1826.90	-	-45.6	10K	1K	1K
2	1444.0	3169.93	-	2434.73	-	-23.2	10K	1K	1K
3	1369.30	5195.94	2893.21	3076.06	-44.3	-40.8	10K	1K	1K
4	1572.80	5619.93	2020.23	2115.83	-64.1	-62.3	10K	1K	1K
5	1534.01	3578.17	-	2037.57	-	-43.1	10K	1K	1K
6	1498.70	3549.97	-	2567.80	-	-27.6	10K	1K	1K
7	1310.01	2300.17	1900.21	1920.10	-17.4	-16.5	10K	1K	1K
8	1484.65	4480.99	2586.38	2661.74	-42.2	-40.6	10K	1K	1K
9	1325.19	3718.86	2310.29	2470.46	-37.9	-33.5	10K	1K	1K
10	1491.49	4123.0	-	2054.80	-	-50.1	10K	1K	1K
11	1364.63	2355.44	-	1828.87	-	-22.3	10K	1K	1K
12	1579.31	3916.14	-	2079.40	-	-46.9	10K	1K	1K
13	1392.41	2792.44	2202.43	2217.17	-21.1	-20.6	10K	1K	1K
14	1437.43	3142.67	-	2279.83	-	-27.4	10K	1K	1K
15	1456.57	4774.58	-	2101.69	-	-55.9	10K	1K	1K
16	1609.06	3402.29	-	2379.86	-	-30.1	10K	1K	1K
17	1499.23	3637.19	-	2086.51	-	-42.6	10K	1K	1K
18	1322.51	3410.98	2108.29	2148.98	-38.2	-37.0	10K	1K	1K
19	1470.73	4591.18	-	2387.17	-	-48.1	10K	1K	1K
20	1443.85	3376.5	-	1967.0	-	-41.7	10K	1K	1K

H2). In particular, the 'Ratio' of H2 for the problems in Table 6 is surprising, reducing the CPLEX upper bound up to -84.2%. Also, note that we find some test problems such that the solution quality of H2 is better than that H1 finds (see, #9~#18 and #20 in Table 5, and #1~#4, #6~#12 and #16~#20 in Table 6). It seems that this is due to the slow-down of solving L_SS($x^U + x^D, f$) in H1 due to the increase of problem size. That is, for larger problems the computing time required to optimally solve L_SS($x^U + x^D, f$) may increase exponentially,

Table 5. Comparison of upper bounds ($|N| = 10$, $|K| = 6$, $L = 7$, $T = 7$ days).

No	LP bound	Upper bound			Ratio		Elapsed Time		
		L_SS	H1	H2	H1	H2	L_SS	H1	H2
1	1188.78	2176.86	2202.12	2221.37	1.2	2.1	10K	1K	1K
2	1115.52	1686.84	1679.23	1802.49	-0.5	6.8	10K	1K	1K
3	1205.39	2502.58	2020.46	2046.47	-19.2	-18.2	10K	1K	1K
4	1145.48	2168.92	1889.70	1940.97	-12.8	-10.5	10K	1K	1K
5	1183.18	2807.32	2245.19	2279.66	-20.1	-18.8	10K	1K	1K
6	1145.67	2168.67	2184.32	2271.48	0.7	4.7	10K	1K	1K
7	1281.12	2839.53	2494.22	2534.32	-12.1	-10.8	10K	1K	1K
8	1290.98	1979.72	2027.21	2096.09	2.4	5.9	10K	1K	1K
9	1150.17	2333.40	2338.27	2293.33	0.2	-1.7	10K	1K	1K
10	1087.74	2073.13	1902.22	1897.78	-8.2	-8.5	10K	1K	1K
11	1196.45	1725.67	1774.78	1737.50	2.8	0.7	10K	1K	1K
12	1174.91	2306.67	2096.68	2075.30	-9.1	-10.1	10K	1K	1K
13	1211.91	2088.45	2003.34	1995.40	-4.1	-4.5	10K	1K	1K
14	1136.06	2110.70	2130.29	2057.24	0.9	-2.5	10K	1K	1K
15	1147.61	2167.85	2140.20	2021.50	-1.3	-6.8	10K	1K	1K
16	1162.16	2603.97	1950.57	1901.18	-25.1	-26.9	10K	1K	1K
17	1193.04	1840.32	1977.23	1959.25	7.4	6.4	10K	1K	1K
18	1104.82	1843.69	1883.34	1866.66	2.1	1.3	10K	1K	1K
19	1324.35	2322.98	2231.23	2233.92	-3.9	-3.8	10K	1K	1K
20	1245.23	2170.56	2044.83	2041.56	-5.8	-5.9	10K	1K	1K

Table 6. Comparison of upper bounds ($|N| = 10$, $|K| = 6$, $L = 15$, $T = 15$ days).

No	LP bound	Upper bound			Ratio		Elapsed Time		
		L_SS	H1	H2	H1	H2	L_SS	H1	H2
1	2152.88	16769.16	4133.57	3652.97	-75.4	-78.2	10K	1K	1K
2	2230.30	10091.81	4776.34	4299.68	-52.6	-57.4	10K	1K	1K
3	2471.71	13853.36	4932.12	4639.17	-64.4	-66.5	10K	1K	1K
4	2231.65	12036.20	5103.01	4103.01	-57.6	-65.9	10K	1K	1K
5	2335.15	8634.32	3977.64	3986.38	-53.9	-53.8	10K	1K	1K
6	2197.34	12788.84	5386.73	4085.34	-57.8	-68.1	10K	1K	1K
7	2314.63	7154.04	4764.47	4065.86	-33.4	-43.2	10K	1K	1K
8	2482.98	6833.23	4823.23	4534.91	-29.4	-33.6	10K	1K	1K
9	2353.45	7394.23	-	3894.41	-	-47.3	10K	1K	1K
10	2516.22	9823.23	7433.22	4189.75	-24.3	-57.4	10K	1K	1K
11	2217.54	23276.51	4833.21	4390.30	-79.2	-81.2	10K	1K	1K
12	2307.11	10718.67	5322.24	4196.94	-50.3	-60.8	10K	1K	1K
13	2257.91	7332.11	4233.57	4251.34	-42.3	-42.1	10K	1K	1K
14	2366.61	28283.43	4468.56	4470.84	-84.2	-84.2	10K	1K	1K
15	2275.01	7702.13	4502.23	4523.02	-41.6	-41.3	10K	1K	1K
16	2337.92	12980.64	-	4577.16	-	-64.7	10K	1K	1K
17	2364.39	10404.48	4983.56	4475.88	-52.1	-56.9	10K	1K	1K
18	2347.34	17659.31	5783.54	5434.53	-67.3	-69.2	10K	1K	1K
19	2312.83	10511.11	-	4175.60	-	-60.3	10K	1K	1K
20	2522.76	10726.39	4572.29	4292.56	-57.4	-59.9	10K	1K	1K

which reduces the chance to investigate a number of different setup configurations for a given time limit. While, it seems that the complexity of optimally solving L_SS(x^U, x^D, f) handled in H2 is manageable for real-size problems.

5. CONCLUSION

In this paper, we addressed a lot-sizing and scheduling problem, which can be used to build a production plan at some systems having state dependent setup time, for example, metal-alloy production systems. Also, we considered setup carry-over and setup overlapping as well as demand splitting. We have developed a mathematical formulation for this problem, of which the size does not increase even if we divide a time period into a number of micro time periods. To find a feasible solution of good quality within reasonable time bound, we developed an efficient heuristic algorithm by devising a decomposition scheme coupled with a local search procedure. Test results show that the developed heuristic algorithm finds good quality feasible solutions using far less computation time compared with the CPLEX.

REFERENCES

Belvaux, G. and Wolsey, L. A. (2000), A specialized branch-and-cut system for lot-sizing problems, *Management Science*, **46**(5), 724-738.

Belvaux, G. and Wolsey, L. A. (2001), Modeling practical lot-sizing problems as mixed- integer programs, *Management Science*, **47**(7), 993-1007.

Clark, A. R. and Clark, S. J. (2000), Rolling-horizon lot-sizing when set-up times are sequence-dependent, *International Journal of Production Research*, **38** (10), 2287-2307.

Drexl, A. and Haase, K. (1995), Proportional lotsizing and scheduling, *International Journal of Production Economics*, **40**(1), 73-87.

Drexl, A. and Haase, K. (1996), Sequential-analysis based randomized-regret-methods for lot-sizing and scheduling, *Journal of the Operational Research Society*, **47**(2), 251- 265.

Gopalakrishnan, M. (2000), A modified framework for modeling set-up carryover in the capacitated lotsizing problem, *International Journal of Production Research*, **38**(14), 3421-3424.

Gupta, D. and Magnusson, T. (2005), The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times, *Computers & Operations Research*, **32**(4), 727-747.

Fleischmann, B. and Meyr, H. (1997), The general lotsizing and scheduling problem, *OR Spektrum*, **19**, 11-21.

Meyr, H. (2000), Simultaneous lotsizing and scheduling by combining local search with dual reoptimization, *European Journal of Operational Research*, **120**(2), 311-326.

Han, J., Lee, Y., Kim, S., and Park, E. (2007), An alternative modeling for lot-sizing and scheduling problem with a decomposition based heuristic algorithm, *Journal of the Korean Institute of Industrial Engineers*, **33**, 373-380.

Meyr, H. (2002), Simultaneous lotsizing and scheduling on parallel machines, *European Journal of Operational Research*, **139**(2), 277-292.

Porkka, P., Vepsalainen, A. P., and Kuula, M. (2003), Multiperiod production planning carrying over set-up time, *International Journal of Production Research*, **41**(6), 1133-1148.

Sox, C. R. and Gao, Y. (1999), The capacitated lot sizing problem with setup carry-over, *IIE Transactions*, **31**(2), 173-181.

Suerie, C. and Stadtler, H. (2003), The capacitated lotsizing problem with linked lot sizes, *Management Science*, **49**(8), 1039-1054.

Suerie, C. (2006), Modeling of period overlapping setup times, *European Journal of Operational Research*, **174**(2), 874-886.