An HNP-MP Approach for the Capacitated Multi-item Lot Sizing Problem with Setup Times

Tao Wu, Leyuan Shi, Senior Member, IEEE, and Neil A. Duffie

Abstract—In this paper we consider the capacitated multi-item lot sizing problem with setup times. The problem is to schedule \( J \) different items over a horizon of \( T \) periods with the objective to minimize the sum of setup cost and inventory holding cost. To achieve feasible high quality solutions, we propose a new solution approach which hybrids Nested Partitions and Mathematical Programming (HNP-MP). Nested Partitions is a partitioning and sampling based heuristic method with a global perspective on the problem. In the proposed new method the Mathematical Programming method is implemented to calculate the promising index and to provide a good guidance on partitioning in the Nested Partitions framework. A time-oriented decomposition heuristic method, Relax-and-Fix, is also implemented to obtain good promising regions and speed up the computational process. Computational results based on benchmark test problems show that the approach is computationally tractable and is able to obtain good results. The approach outperforms other state-of-the-art approaches found in the literature.

Note to Practitioners—The capacitated multi-item lot sizing problem with setup times considered in this paper is motivated by the attempt of effectively planning the production with lower setup cost and inventory holding cost under the constraints of limited capacities. An effective production planning on a lot sizing problem can provide a good guidance on the decision made by a Material Requirements Planning (MRP) system, and check for the conflicts with available capacities. Two areas of approaches are implemented to deal with the problem: the exact approach and the heuristic approach. The main drawback of the exact approach is the excessive computational time when the problem size becomes large. On the other hand, the main drawback of the heuristic approach is an inefficient objective function. The goal of this paper is to propose a heuristic approach which outperforms other state-of-the-art approaches found in the literature. Practitioners can apply the proposed method into the extensions of lot sizing problems which consider backlogging, overtime, and shortage.

Index Terms—Capacitated Multi-item Lot Sizing, Dantzig-Wolfe Decomposition, Nested Partitions, Column Generation, Relax-and-Fix.

I. INTRODUCTION

The paper addresses the capacitated single-level multi-item lot sizing problem with setup times (CLST). The problem is to plan the production of \( J \) items over a horizon of \( T \) periods in which the throughput time is one period. Demands are specified for each item in each period and should be satisfied without backlogging. However, initial inventory is allowed to avoid the infeasibility of a production plan. The objective is to find a minimum cost production plan without violating the capacity constraints in each period. The total cost of any production plan contains three components: setup cost, inventory holding cost and initial inventory holding cost. The following assumptions are addressed in the problem. Setup times and cost are non-sequence dependent. There are no linked lot sizes, e.g. no distinction between major setups for product types and minor setups for items. The possibility of setup carryover between periods is neglected. The investigated model is of the big-bucket type that does not consider the time-phasing of production activities within a period. In particular, the investigated model does not exclude interferences between the production activities at different machines. The lot sizing problem reflects the settings of an MRP and control environment based on the push principle. The solution procedures that have been applied to solve the CLST can be generally divided into two areas.

The first area encompasses the exact solution approaches related to Mathematical Programming, which itself contains two groups. The first group strengthens the original formulation by adding valid inequalities. Barany et al. [1] propose valid inequalities that define the polytope of the uncomplicated single-item lot sizing problem. Pochet and Wolsey [2], Wolsey [3], and Miller et al. [4] add strong valid inequalities to the original model formulation to speed up solving these lot sizing problems. The second group within this method attempts to create strong reformulations. Two of the most prominent reformulation models are the Facility Location reformulation and the Shortest Path reformulation proposed by Krarup and Bilde [5] and Eppen and Martin [6], respectively. Stadler [7] proposes another reformulation model. However, experience shows that exact solution approaches cannot handle large mathematical mixed-integer programming problems because of unacceptably long solution times; only small problem instances can be solved within reasonable time limits.

The other area of solution procedures involves heuristic methods, such as the neighborhood search, item oriented decomposition, and time-oriented decomposition approaches. Such methods are more often implemented in research papers than exact solution approaches because even the capacitated single-item lot sizing problem has been shown to be NP-hard by Florian et al. [8]. Belvaux and Wolsey [9] propose a method, which is a specialized Branch-and-Cut and time-oriented decomposition heuristic method for lot sizing prob-
lems. Federgruen et al. [10] develop and analyze a class of so-called progressive interval heuristics to the CLST. In addition, Degraeve and Jans [11] describe an implementation of a Branch-and-Price algorithm to the CLST. A combination of simplex optimization and subgradient updating is used to speed up the Column Generation process.

In other heuristic implementations, Trigeiro et al. [12] and Tempelmeier and Derstroff [13] propose Lagrangian based decomposition methods for the large scale lot sizing problem. Salomon [14] develops a tabu search procedure where the solution set is searched quickly in a greedy fashion. Simpson and Erenguc [15] develop a neighborhood search heuristic method. Sahling et al. [16] propose an MP-based fix-and-optimize approach that solves a series of mixed-integer linear programs iteratively for the dynamic multi-level capacitated lot sizing problem with setup carry-overs. Abisi et al. [17] propose mixed integer programming heuristics based on a planning horizon decomposition strategy to find a feasible solution for the multi-item capacitated lot-sizing problem with setup times and shortage costs. Pochet et al. [18] develop a library of tools for solving production planning problems that is called as LS-LIB. LS-LIB is embedded into an optimization software, XPRESS, and is very powerful to solve production planning problems. The strength of the heuristic methods is their ability to obtain the fairly good feasible solutions or upper bounds within a reasonable time frame. However, the duality gap between the upper bound and lower bound can be unacceptable because the upper bounds achieved by the heuristic methods are insufficient. We therefore propose a new solution approach which is called the Hybrid Nested Partitions and Mathematical Programming approach (HNP-MP).

Nested Partitions is a partitioning and sampling based method that focuses computational effort on the most promising region of the solution space while maintaining a global perspective on the problem [19]. It has been efficiently applied to a wide range of Mixed Integer Programming problems such as the traveling salesman problem [20], the product design problem [21], and the local pickup and delivery problem [22]. In the Nested Partitions framework, calculating the promising index of each region is critical to having the fast convergence rate and obtaining good feasible solutions. Nested Partitions could be implemented to the CLST directly. However, it would be hard to find a good partitioning and sampling strategy if it is just used individually. Here, the lower bounds found using Column Generation or Simplex and upper bounds found using a time-oriented decomposition heuristic method, Relax-and-Fix, are used to define intelligent partitioning that improves the efficiency of the Nested Partitions method. The solutions in both upper bounds and lower bounds are utilized and provide guidance on partitioning and sampling in Nested Partitions. Also, the efficiency of the method could be achieved by setting a comparatively small mount of time to each subproblem for the Relax-and-Fix algorithm; it then would not consume much computational effort.

The remainder of the paper is organized as follows: A Facility Location model formulation for the CLST is given in Section II. In Section III, a Dantzig-Wolfe Decomposition reformulation of the Facility Location model is described and a Column Generation algorithm is proposed to solve its Linear Programming (LP) relaxation problem. Nested Partitions and Relax-and-Fix are also introduced in this section. After that the HNP-MP approach is proposed. In Section IV, a further extension analysis of HNP-MP is given. Detailed computational tests based on benchmark problems are given in Section V. Finally, our conclusions and future research are presented in Section VI.

II. THE MODEL FORMULATION FOR THE CLST

Several model formulations yielding different model sizes and lower bounds have been proposed for the CLST in the literature. We formulate the problem as a Facility Location problem, which was originally proposed by Krarup and Bilde [5] for single-item problems. The Facility Location formulation is able to provide a strong lower bound when its LP relaxation problem is solved. In order to present the model formulation clearly, notations are given as follows:

Indices and Index Sets:

- \( T \) Number of periods.
- \( J \) Number of items.
- \( M \) Number of machines.
- \( t, s \) Periods, \( t, s = 1, \ldots, T \).
- \( j \) Items, \( j = 1, \ldots, J \).
- \( m \) Machines, \( m = 1, \ldots, M \).

Parameters:

- \( sc_j \) Setup cost for a lot of item \( j \).
- \( ST_{jm} \) Setup time for item \( j \) on machine \( m \).
- \( C_{mt} \) Available capacity of machine \( m \) in period \( t \).
- \( hc_j \) Inventory holding cost for one unit of item \( j \) in a period.
- \( gc_{jt} \) Unit cost of initial inventory for item \( j \) to be used in period \( t \).
- \( a_{jm} \) Production time necessary to produce one unit of item \( j \) on machine \( m \).
- \( d_{jt} \) Demand for item \( j \) in period \( t \).

Variables:

- \( X_{jts} \) The amount of item \( j \) produced in period \( t \) to meet demand in period \( s (t \leq s) \).
- \( I_{jt} \) The amount of initial inventories of item \( j \) to meet demand in period \( t \), these variables are similar to lost sales or shortages known in the literature [23] or [17].
- \( Y_{jt} \) Binary setup variable (=1, if item \( j \) is produced in period \( t \); 0 otherwise).

The formulation of the problem, which we refer to as the Facility Location (FL) model formulation, can be then stated as follows:

\[
\min \sum_{j=1}^{J} \sum_{t=1}^{T} sc_j Y_{jt} + \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{s=t}^{T} (s-t) hc_j X_{jts}
\]
\( + \sum_{j=1}^{J} \sum_{t=1}^{T} g_{jt} I_{jt}. \) \hspace{1cm} (1)

\[
\begin{align*}
\sum_{t=1}^{a} X_{jts} + I_{js} & = d_{js} \quad \forall j = 1, \ldots, J, \ s = 1, \ldots, T. \hspace{1cm} (2) \\
\sum_{j=1}^{J} \sum_{s=1}^{T} a_{jm} X_{jts} + \sum_{j=1}^{J} S_{tjm} Y_{jt} & \leq C_{mt} \\
& \quad \forall m = 1, \ldots, M, \ t = 1, \ldots, T. \hspace{1cm} (3) \\
X_{jts} & \leq d_{js} Y_{jt} \quad \forall j = 1, \ldots, J, \ t = 1, \ldots, T, \ s = t, \ldots, T. \hspace{1cm} (4) \\
I_{jt} & \geq 0; \ X_{jts} \geq 0; \ Y_{jt} \in \{0,1\} \quad \forall j = 1, \ldots, J, \ t, s = 1, \ldots, T. \hspace{1cm} (5)
\end{align*}
\]

The objective function (1) minimizes total setup, inventory holding, and initial inventory holding costs during the planning horizon. Constraints (2) ensure demand satisfaction in all periods for all items. Constraints (3) enforce capacity requirements. Constraint set (4) ensures that the total amount of production is less than sufficiently large value and that a setup is performed in period \( t \) for item \( j \) if any demand is satisfied using the corresponding production. Constraints (5) enforce the binary and nonnegative requirements for different variables.

III. THE HNP-MP APPROACH FOR THE CLST

In order to present the HNP-MP approach clearly, we firstly make a brief introduction to the Nested Partitions method in this section. A Dantzig-Wolfe Decomposition reformulation of the FL model is presented and a Column Generation algorithm is implemented to solve its LP relaxation problem, which provides a strong lower bound and a guideline on calculating the promising index for HNP-MP. A Relax-and-Fix method is also implemented, it helps find a good promising region and improves the computational speed for HNP-MP. Finally, we make a detailed explanation of the implementation of HNP-MP.

A. Nested Partitions

The Nested Partitions method is a partitioning and sampling based strategy that focuses computational effort on the most promising region of the solution space while maintaining a global perspective on the problem. In each iteration of the algorithm, the entire solution space is viewed as the union of a promising region and a surrounding region. The actual Nested Partitions iteration comprises four steps, which we outline below:

The first step of each iteration is to partition the current most promising region into several subregions and aggregate the surrounding region into one region. The partitioning strategy imposes a structure on the feasible region and is therefore very important for the convergence algorithm. If the partitioning is such that most of the good solutions to the problem tend to be clustered together in subregions, it is likely that the algorithm quickly concentrates the search in these subsets of the feasible region. It should be noted that a good partitioning strategy always exists but may not be easy to identify.

The next step of the algorithm is to randomly sample from each of the subregions and from the aggregated surrounding region. This can be done in almost any fashion. The only condition is that each solution in a given sampling region should be selected with a positive probability. Clearly, uniform sampling can always be chosen as the generic option. However, it may often be worthwhile to incorporate a special structure into the sampling procedure. The aim of such a sampling method should be to select good solutions with a higher probability than poor solutions.

Once each region has been sampled, the next step is to use the sample points to calculate the promising index of each region. Again, the Nested Partitions methodology offers a great deal of flexibility. The only requirement imposed on a promising index is that it agrees with the original performance function on regions of maximum depth, i.e., on singletons that are regions containing only a single point. Due to such a simple requirement, many local search heuristics can be used in this step to construct a promising index [19].

If one of the subregions has the best promising index, the algorithm moves to this region and considers it to be the most promising region in the next iteration. If the surrounding region has the best promising index, the algorithm backtracks. This can be done in several manners; two of which will be discussed here. The first alternative considered is backtracking all the way to the top, i.e., to the entire feasible region. The second is to backtrack to the superregion of the current most promising region. Both of these backtracking rules ensure asymptotic convergence. [19] can be referred for more information on Nested Partitions.

B. The Dantzig-Wolfe Decomposition and Column Generation Approach for the CLST

The Dantzig-Wolfe Decomposition approach is an application of using a pricing mechanism that has been developed for a wide class of mathematical programs, especially for those problems with complicating or linking constraints. It chooses to solve a large number of smaller size, typically well-structured, subproblems instead of solving the original problem whose size and complexity are beyond what can be solved within a reasonable amount of time. Also, the lower bound provided by the relaxation problem of Dantzig-Wolfe Decomposition is stronger than that obtained from the relaxation of the original problem. For the CLST, the original problem is here separated into a master problem and \( J \) subproblems in the Dantzig-Wolfe Decomposition method. All generated columns in the master problem are the vertex points of the convex space defined by the constraints in \( J \) subproblems. The master problem has capacity constraints and convexity constraints for all columns. While \( J \) subproblems are independent uncapacitated single item lot sizing problems.
The Dantzig-Wolfe Decomposition approach for the CLST is addressed by a number of authors including Vanderbeck et al. [24] and Degraeve and Jans [11]. Most of them deal with the LP relaxation of the master problem in order to obtain a lower bound. In this paper we implement the Dantzig-Wolfe Decomposition into the FL model formulation (FL-DD) because it provides a strong lower bound. It is similar to those used by Manne [25] and Degraeve and Jans [11]. However, the lower bounds achieved by the Dantzig-Wolfe Decomposition used in the paper are better than theirs because they implement the Dantzig-Wolfe Decomposition on a weak formulation. Please refer [25] for more details. Let \( q \in Q^j \) where \( Q^j \) is defined as the set of all possible setup schedules for item \( j \), \( Q^j_t = (Y^j_{1t}, \ldots, Y^j_{Tt}) | Y^j_{it} \in \{0,1\} \forall t \in 1,\ldots, T \). \( Y^j_{it} \) is one if there is a setup for item \( j \) in period \( t \) in setup schedule \( q \); zero otherwise. \( X^q_{jts} \) is defined as the amount of item \( j \) produced in period \( t \) to meet the demand in period \( s \) (\( t \leq s \)) in setup schedule \( q \), \( I^q_{jt} \) is defined as the initial inventories of item \( j \) to meet demand in period \( t \) in setup schedule \( q \), and \( z_{jq} \) is defined as the fraction of schedule \( q \) for product \( j \) that will be produced. The master problem formulation of FL-DD, which is referred to as FLM, is then given as follows:

**FLM:**

\[
\begin{align*}
\min & \quad \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{q \in Q^j} s c_j Y^q_{jts} z_{jq} + \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{s \in Q^j} (s-t) h c_j X^q_{jts}, \\
\text{s.t.} & \quad \sum_{q \in Q^j} z_{jq} = 1 \quad \forall j = 1, \ldots, J \\
& \quad \sum_{j=1}^{J} \sum_{t=1}^{T} a_{jm} X^q_{jts} z_{jq} + \sum_{j=1}^{J} \sum_{q \in Q^j} S T_j Y^q_{jts} z_{jq} \leq C_{mt} \\
& \quad \forall m = 1, \ldots, M, t = 1, \ldots, T \\
& \quad z_{jq} \geq 0 \quad \forall j = 1, \ldots, J, q \in Q^j
\end{align*}
\]

For the CLST, Column Generation starts with a restricted master problem with only a few columns obtained from the uncapacitated single-item lot sizing pricing problem. Therefore, the initial restricted master problem FLM may be infeasible because the integrality capacity constraints may not be satisfied. In order to get a feasible initial FLM, we can either use a heuristic method to generate some feasible initial columns in FLM, or slightly modify FLM by adding an artificial variable to the right-hand side of each of the integrality capacity constraints (**8**) and considering this artificial variable as the objective function. The second technique, adding an artificial variable, is used in this paper, and the problem is referred to as being in Phase1 when the added artificial variable is positive. Note that the feasible polyhedron for the lot sizing problem is bounded because the ending inventory must be zero. Otherwise, a linear combination of extreme rays needs also to be considered. The master problem FLM1 and the pricing subproblem FLS1 in Phase1 are stated as follows. Note that the artificial variable \( O_f \) is introduced in Phase1 to avoid infeasible initial columns. Below we will use \( \pi_{mt} \) \((m = 1, \ldots, M, t = 1, \ldots, T)\) as the dual variables associated with the joint capacity constraints (**11**), and \( u_j \) \((j = 1, \ldots, J)\) as dual variables for the second set of constraints (**12**), known as convexity constraints.

**FLM1:**

\[
\begin{align*}
\min & \quad O_f \\
\text{s.t.} & \quad \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{m=1}^{M} a_{jm} X^q_{jts} z_{jq} + \sum_{j=1}^{J} \sum_{q \in Q^j} S T_j Y^q_{jts} z_{jq} \leq C_{mt} \\
& \quad + O_f \quad \forall t = 1, \ldots, T, m = 1, \ldots, M \\
& \quad \sum_{q \in Q^j} z_{jq} = 1 \quad \forall j = 1, \ldots, J \\
& \quad z_{jq} \geq 0 \quad \forall j = 1, \ldots, J, q \in Q^j
\end{align*}
\]

**FLS1:**

\[
\begin{align*}
\max & \quad \sum_{t=1}^{T} \sum_{m=1}^{M} a_{jm} \pi_{mt} X^q_{jts} + \sum_{t=1}^{T} \sum_{m=1}^{M} S T_j m \pi_{mt} Y^q_{jts} \\
& \quad + u_j \quad \forall j = 1, \ldots, J \\
\text{s.t.} & \quad (**2**), (**4**), (**5**) \nonumber
\end{align*}
\]

When the optimal solution of \( O_f \) in FLM1 is non-positive after plugging more and more columns into the master problem FLM1. The problem is moved to Phase2, in which the master problem FLM2 and the pricing subproblem FLS2 are given as follows. Note that \( O_f \) is set as 0 in Phase2.

**FLM2:**

\[
\begin{align*}
\min & \quad \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{s=t}^{T} (s-t) h c_j X^q_{jts} z_{jq} + \sum_{j=1}^{J} \sum_{q \in Q^j} \sum_{t=1}^{T} s c_j Y^q_{jts} z_{jq} \\
& \quad \sum_{j=1}^{J} \sum_{q \in Q^j} \sum_{t=1}^{T} g c_j I^q_{jts} z_{jq} \\
\text{s.t.} & \quad (**11**), (**12**), (**13**) \nonumber
\end{align*}
\]

**FLS2:**

\[
\begin{align*}
\max & \quad \sum_{t=1}^{T} \sum_{s=t}^{T} \sum_{m=1}^{M} a_{jm} \pi_{mt} X^q_{jts} + \sum_{t=1}^{T} \sum_{m=1}^{M} S T_j m \pi_{mt} Y^q_{jts} \\
& \quad + u_j - \sum_{t=1}^{T} s c_j Y^q_{jts} - \sum_{t=1}^{T} \sum_{s=t}^{T} (s-t) h c_j X^q_{jts} \\
& \quad - \sum_{t=1}^{T} g c_j I^q_{jts} \quad \forall j = 1, \ldots, J
\end{align*}
\]
At each iteration of the Column Generation algorithm in Phase2, each uncapacitated single-item subproblem FLS2 for each item $j$ is solved to minimize the reduced cost. Columns with a negative reduced cost are added to the master problem FLM2. Next, the master problem is solved again as an LP problem and try to find new columns that price out with the new dual prices in a new iteration. If any new column with a negative reduced cost can not be found, the optimal solution of the restricted master problem is the final optimal solution for the original problem.

Several techniques can be implemented to accelerate the Column Generation process. If more than one column with a negative reduced cost is available from the subproblem solution, it may be beneficial to add multiple columns, instead of only the column with the most negative reduced cost, to the restricted master problem. In addition, the subproblems are independent, so a parallel-computing environment can be implemented to solve subproblems, accelerating the solving process. The Column Generation algorithm is follows:

**Column Generation algorithm**

**Step1:** Solve $J$ subproblems FLS2 in Phase1 by setting the dual multipliers $\pi_{mt}, u_j$ to zero, and then plug solutions into the restricted master problem FLM1 as the first $J$ initial columns. Go to Step2.

**Step2:** Solve the restricted master problem FLM1. If the optimal solution of $O_f$ is nonpositive, go to Step4; otherwise go to Step3.

**Step3:** Get $J$ subproblems FLS1 based on the dual multipliers of the constraints of FLM1, and then solve them to plug more columns into FLM1. Go to Step2.

**Step4:** Set the artificial variable $O_f$ to zero in Phase2, and then solve the master problem FLM2 to be optimal. Go to Step5.

**Step5:** Get $J$ subproblems FLS2 based on the constraints dual multipliers of FLM2, and then solve them to plug more columns into FLM2. If all optimal solutions of $J$ subproblems FLS2 are nonnegative, stop the algorithm; otherwise go to Step4.

C. The Relax-and-Fix Algorithm for the CLST

Relax-and-Fix is a time-oriented decomposition heuristic method well-suited to the lot-sizing problem. It is a relatively simple and straightforward approach for decomposing the original MILP lot-sizing problem into several smaller MILP problems by fixing the binary variables in some periods and relaxing the binary variables so they are continuous in the remaining periods. Belvaux and Wolsey [9] propose a branch-and-cut system that is the first production planning tool using this technique. Stadtler [26] proposes another heuristic approach based on Relax-and-Fix, which uses internally rolling schedules with overlapping lot-sizing windows and relaxed integrality constraints.

The Relax-and-Fix method we attempt to apply is similar to but much simpler than the Stadtler’s Heuristic method. In this method, the predefined length $\alpha$ of periods is chosen as "time window", and $\beta$ indicates the number of periods that do not overlap with the next window. We solve the problem by using the technique proposed by Belvaux and Wolsey [9], but the difference is that we only fix the binary variables for $\beta$ periods that do not overlap with the next window.

To present the sub-model formulations used in the Relax-and-Fix algorithm clearly, we introduce three sets for the setup decision variables: TF which is the set of the binary setup variables $Y_{jt}$ fixed in the previous iterations; TI which is the set of the binary setup variables $Y_{jt}$ that remain to be binary; and TR which is the set of the temporary LP relaxations of the setup variables $Y_{jt}$. We also assume that the setup variables in the sets TF, TI, and TR are $Y_{jt}^{tf}, Y_{jt}^{ti}$ and $Y_{jt}^{tr}$, respectively. The sum of TF, TI, and TR is the full set for the setup decision variables. The sub-model names of FL are given as SubFL and its formulation is stated as follows. Note that, in the formulation, new variables $Z_{jt}$, the summative variables of $Y_{jt}^{tf}, Y_{jt}^{ti}$ and $Y_{jt}^{tr}$, are introduced.

**SubFL:**

\[
\begin{align*}
\min & \sum_{j=1}^{J} \sum_{t=1}^{T} sc_j Z_{jt} + \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{s=1}^{T} (s-t)hc_j X_{jts} \\
& + \sum_{j=1}^{J} \sum_{t=1}^{T} gc_j I_{jt}. \quad (17)
\end{align*}
\]

s.t.

\[
Z_{jt} = Y_{jt}^{tf} + Y_{jt}^{ti} + Y_{jt}^{tr}, \quad \forall j = 1, \ldots, J, t = 1, \ldots, T. \quad (18)
\]

\[
\sum_{t=1}^{s} X_{jts} + I_{js} = d_{js}, \quad \forall j = 1, \ldots, J, t = 1, \ldots, T. \quad (19)
\]

\[
\sum_{j=1}^{J} \sum_{s=1}^{T} a_{jm} X_{jts} + \sum_{j=1}^{J} ST_{jm} Z_{jt} \leq C_{mt}, \quad \forall m = 1, \ldots, M, t = 1, \ldots, T. \quad (20)
\]

\[
X_{jts} \leq d_{js} Z_{jt}, \quad \forall j = 1, \ldots, J, t = 1, \ldots, T, \quad s = t, \ldots, T. \quad (21)
\]

\[
I_{jt} \geq 0; \quad X_{jts} \geq 0; \quad Y_{jt}^{ti} \in \{0, 1\}; \quad 0 \leq Y_{jt}^{tr} \leq 1, \quad \forall j = 1, \ldots, J, t, s = 1, \ldots, T. \quad (22)
\]

In the above sub-models, TF is the set including all setup variables in the beginning periods of the problem which have already been fixed as binary values during previous iterations. We assume all of the period numbers form the set TIP and $|\text{TIP}|$ is the size of TIP. For example, if TF is the set including all setup variables from periods 1 to 4, then TFP is a set which includes 1, 2, 3, and 4, and $|\text{TIP}|$ equals to 4. TI is the set including those in $\alpha$ subsequent periods, and all of these period numbers are assumed to form the set TIP and $|\text{TIP}|$ is the size of TIP. While TR is the set including those in the
remaining periods, we assume the period numbers form the set TRP and $|TRP|$ is the size of TRP. After that the Relax-and-Fix algorithm can be stated as follows. Note that, in the algorithm, all period numbers that are larger than $T$ are invalid and will not be input into the corresponding set.

**The Relax-and-Fix Algorithm**

Set the iteration number, $\textit{iter} = 0$, and TFP as an empty set. do

- Set $\textit{TIP} = \{|\text{TFP}| + 1, \ldots, |\text{TFP}| + \alpha\}$. Set $\textit{TRP} = \{|\text{TFP}| + \alpha + 1, \ldots, T\}$ (If $|\text{TFP}| + \alpha + 1 > T$, TRP is an empty set).
- Solve SubSE-I&L by using an MIP solver in limited time, and then obtain the solutions of $y_{it}$.
- Set $\textit{iter} = \textit{iter} + 1$.
- Set TFP = $\{1, \ldots, \textit{iter} \cdot \beta\}$.
- Fix all binary solutions into setup variables $y_{it}$ if $t \in$ TFP, while ($\text{TFP} \neq \emptyset$).

**D. The Detailed Analysis of the HNP-MP Algorithm**

To present the HNP-MP algorithm clearly, we consider an illustrative example here that plans the production of 1 item over a horizon of 6 periods. In the example there are a finite number of continuous variables and only 6 binary setup decision variables, $Y_{11}, Y_{12}, \ldots, Y_{16}$, so the finite solution set $\Theta$ for all binary variables contains all enumeration alternatives of 6 binary variables with binary values 0 or 1. Our purpose is to find a point with the best performance within the enumeration alternatives. Because of the special characteristics of the CLST, we decide to fix setup decision variables periods by periods in the HNP-MP algorithm. In this trivial example, let us assume we plan to fix 3 periods of binary variables in each iteration, this is to say that we fix all 6 binary variables with two iterations. $Y_{11}, Y_{12}, Y_{13}$ are fixed in the first iteration, and $Y_{14}, Y_{15}, Y_{16}$ are fixed in the second.

In the first iteration, nothing is assumed to be known about where good solutions are, hence we use the entire feasible region $\Theta$ as the most promising region. The Dantzig-Wolfe Decomposition is applied to the LP relaxation of the FL formulation, and then the Column Generation algorithm is applied to achieve the LP relaxation solutions, which are referred to as Linear Solutions. Meanwhile, the Relax-and-Fix algorithm is also implemented to solve the FL formulation to achieve feasible binary solutions, which are referred to as Binary Solutions. The Linear Solutions and the Binary Solutions of the binary variables $Y_{11}, Y_{12},$ and $Y_{13}$ are then compared, the variables are fixed with the solution values if they are the same. The binary variables having the same solutions are referred to as EquiSolution Variables. Let us assume the solutions of $Y_{11}$ are the same and $Y_{11}$ is the EquiSolution Variable in the example shown in Figure 1. Under such conditions, the Promising Region is the region in which $Y_{11}$ is fixed with the value of 1 and other binary variables are arbitrary but satisfying constraints, which is a subset of $\Theta$. The remaining region in the set $\Theta$ is considered as the Surrounded Region. Because we need to fix $Y_{11}, Y_{12},$ and $Y_{13}$ in the first iteration and the EquiSolution Variable $Y_{11}$ has already been fixed, we then only need to sample the binary variables $Y_{12}$ and $Y_{13}$ in the Promising Region. Such variables are referred to as Partitioning Variables.

Let us assume we do sampling on $Y_{12}$ and $Y_{13}$ for 50 times in the Promising Region to get 50 sample problems. A weighted sampling strategy is used here to improve the quality of samples. It utilizes both of Linear Solutions and Binary Solutions. If the summed solution value of a binary variable is found to be larger than 1, it will then have a higher probability, say 0.65, to be selected as 1 in sampling; if the total value is smaller than 1, the binary variable will be then selected as 1 in sampling with a lower probability, say 0.35. Otherwise the binary variable will be randomly selected as 0 or 1 with the same probability. Each of these sample problems is derived from the FL formulation where $Y_{11}$ is fixed as a constant with the binary value of 1, and $Y_{12}$ and $Y_{13}$ are temporarily fixed into the sampling values. Other 5 sample problems are obtained by sampling on $Y_{11}, Y_{12},$ and $Y_{13}$ for 5 times in the Surrounded Region. Each of these 5 problems is derived from the FL formulation where $Y_{11}, Y_{12},$ and $Y_{13}$ are temporarily fixed into the sampling values. The Dantzig-Wolfe Decomposition method and the Column Generation algorithm are then applied to solve each of the LP relaxation problems of these 55 sample problems to get lower bounds. The lower bound provides guidance for the next Promising Region because the sample problem with a better lower bound is more promising to achieve a better binary solution. Therefore, in this example, we pick out 4 sample problems with better lower bounds within these 55 problems, they are solved by using the Relax-and-Fix algorithm. The sample problem achieving the best solution is then considered as the most promising sample. The Partitioning Variables $Y_{12}$ and $Y_{13}$ in this sample problem will be fixed if the sample problem is from the Promising Region; otherwise we backtrack. It is assumed here that the best sample problem is in the Promising Region and the fixed values of $Y_{12}$ and $Y_{13}$ are 0 and 1. Finally the binary variables $Y_{11}, Y_{12},$ and $Y_{13}$ are fixed as 1, 0, and 1, respectively, in the first iteration and the Promising Region now becomes the region in which $Y_{11}, Y_{12},$ and $Y_{13}$ are fixed as 1, 0, and 1 and other integer variables are arbitrary but satisfying constraints. The remaining region in the set $\Theta$ is then the new Surrounded Region.

In the second iteration, we solve the FL formulation where $Y_{11}, Y_{12},$ and $Y_{13}$ have already been fixed as 1, 0, and 1 using the Column Generation algorithm and the Relax-and-Fix algorithm. Let us assume, in this example, that $Y_{14}$ is the EquiSolution Variable having the same solution value of 0, and the variables $Y_{15}$ and $Y_{16}$ are the Partitioning Variables. Therefore, the Promising Region becomes the region in which $Y_{11}, Y_{12}, Y_{13},$ and $Y_{14}$ are fixed as 1, 0, 1 and 0, and other variables are arbitrary but satisfying constraints. We only do sampling on $Y_{15}$ and $Y_{16}$ in the Promising Region, and do sampling on $Y_{11}, Y_{12}, \ldots,$ and $Y_{16}$ in the Surrounded Region. Then similar procedure needs to be done as we did in the first iteration. Let us assume the sample problem with sampling values 1 and 0 of $Y_{15}$ and $Y_{16}$ finally has the best feasible solution, then the final solution of binary variables $Y_{11}, Y_{12}, Y_{13}, Y_{14}, Y_{15},$ and $Y_{16}$ is 1, 0, 1, 0, 1, and 0, respectively. Once all binary variables are fixed, the final solution of $X_{jts}$
The HNP-MP Algorithm

Step0: Initialization. Set the depth of iteration $\ell = 1$, the initial Promising Region as the entire feasible solution space $\Theta$, and the initial Surrounding Region as an empty set $\emptyset$.

Step1: Solving PFL(\ell) using an MIP solver. The problem PFL(\ell) is solved using an MIP solver within a limited time, such as 10 seconds. If it can achieve optimality, we save the solution and stop the algorithm; otherwise, go to Step2.

Step2: Obtaining the Promising Region PR(\ell) and the Surrounding Region SR(\ell). The problem PFL(\ell) is solved using the Relax-and-Fix algorithm, and its LP relaxation is solved by applying the Dantzig-Wolfe Decomposition method and the Column Generation algorithm or by using the Simplex method. The EquiSolution Variables for $Y_{f_{\ell}}$ are then known by comparing both of the Linear Solutions and the Binary Solutions. After fixing the EquiSolution Variables, the Promising Region PR(\ell) is then known. It is a subset of $\Theta$ where $Y_{f_1}, Y_{f_2}, \ldots, Y_{f_{\ell-1}}$ and EquiSolution Variables of $Y_{f_{\ell}}$ are fixed as constants, and other binary variables are arbitrary but satisfying constraints. The remaining regions are aggregated together to form the Surrounding Region SR(\ell). Go to Step3.

Step3: Sampling-based partitioning. In Step 2, the Partitioning Variables for $Y_{f_{\ell}}$ are also known by comparing both of the Linear Solutions and the Binary Solutions. We then get $N_1(\ell)$ sample problems from the Promising Region PR(\ell) by sampling the Partitioning Variables. Each of these $N_1(\ell)$ sample problems is derived from PFL(\ell) where the EquiSolution Variables for $Y_{f_{\ell}}$ are temporarily fixed as the sampling values. We also get $N_2(\ell)$ sample problems from the Surrounding Region SR(\ell) by sampling $Y_{f_1}, Y_{f_2}, \ldots, Y_{f_{\ell}}$. Each of these $N_2(\ell)$ sample problems are derived from the FL formulation where $Y_{f_1}, Y_{f_2}, \ldots, Y_{f_{\ell}}$ are temporarily fixed as the sampling values. Go to Step4.

Step4: Estimating the promising index. The $N(\ell)$ sample problems are LP relaxed and solved by applying the Dantzig-Wolfe Decomposition method and the Column Generation algorithm or using the Simplex method to estimate their promising indices. We then pick out $N_3(\ell)$ sample problems having better promising indices than others within these $N(\ell)$ sample problems. They are solved again by using the Relax-and-Fix algorithm. The sample problem achieving the best
solution is then considered as the most promising sample. Also, its solution is saved. If the best sample problem is obtained from the Surounding Region \( SR(\ell) \), go to Step5. Or if the best sample problem is obtained from the Promising Region \( PR(\ell) \), the Partitioning Variables for \( Y_{f_1} \) are fixed as the sampling values. Let \( \ell = \ell + 1 \). Go to step1.

**Step5: Backtracking.** Backtrack to the problem PF(\( \ell - 1 \)) in the previous larger region, and let \( \ell = \ell - 1 \). Go to Step1.

As shown in Figure 2, in the first iteration of the HNP-MP approach, we could apply an LP based Mathematical Programming method to the LP relaxation of the problem PF\( (1) \) in order to achieve Linear Solutions or lower bound solutions. Meanwhile, Relax-and-Fix is also implemented to solve the problem PF\( (1) \) to achieve Binary Solutions or upper bound solutions. The upper bound solutions and the lower bound solutions for \( Y_{f_1} \) are then compared, the variables are fixed with the values of the binary solutions if the solutions are the same. The Promising Region \( PR(1) \) and Surounding Region \( SR(1) \) are obtained according to the solutions, we then do sampling to get \( N_1(1) \) sample problems in the Promising Region, and get \( N_2(1) \) sample problems in the Surrounding Region. An LP based Mathematical programming method is then applied to solve each of the LP relaxations of these sample problems to get promising indices. We pick out \( N_3(1) \) sample problems with better lower bounds within these sample problems, they are solved again by Relax-and-Fix. The sample problem achieving the best solution is considered as the most promising sample. The Partitioning Variables in \( Y_{f_1} \) in this sample problem will be fixed if the sample problem is from the Promising Region; otherwise do backtracking. In the second iteration, the same strategy can be used to fix \( Y_{f_2} \). The problem is solved by doing this iteratively. The algorithm stops when the subproblem is small and easy to be solved into optimality with a small amount of time.

IV. THE EXTENDED ANALYSIS OF THE HNP-MP APPROACH

The approach used to obtain random samples from each region in each iteration is flexible for the Nested Partitions algorithm. The only requirement is that each point in a sampling region should have a positive probability of being selected. Uniform distribution sampling scheme works well in most cases. However, integrating Relax-and-Fix and Column Generation into the sampling scheme can significantly improve the sampling quality for the CLST. In HNP-MP, we implement a weighted solution-based sampling method by utilizing the solutions obtained from the LP based algorithm and the Relax-and-Fix algorithm. In addition, we also utilize the special characteristics of the CLST to obtain good samples. For instance, if \( sc_j \) is comparatively higher than \( h_{c_j} \times d_{j(t+1)} \) for \( t = 1, \ldots, T \), then the setup decision variables of this item in each period will be sparser. Therefore, we will give much higher probability to sample the binary variable \( Y_{j(t+1)} \) to be 0 if we find \( Y_{f_1} \) equals to 1 for such item. In each iteration of the HNP-MP algorithm, we also implement a data structure that is used to store samples, and then we can neglect a sample if it is identical with one of the previous sampled points.

A. Sampling-based Partitioning

Efficient partitioning has an impact on the efficiency of the algorithm because an effective partitioning scheme will keep good solutions clustered in the next Promising Region, and HNP-MP can then quickly identify a set of near optimal solutions. In the \( \ell \)th iteration of the HNP-MP algorithm, we know EquSolution Variables and Partitioning Variables for \( Y_{f_1} \) by comparing Linear Solutions with Binary Solutions. Because the amount of Partitioning Variables of \( Y_{f_1} \) might be large, solving all enumerated possible partitioning problem alternatives will be time consuming.

We therefore obtain and solve a fraction of all possible enumeration problem alternatives by applying a sampling-based partitioning strategy. Random sampling-based partitioning might produce efficient sample problems. However, applying a special strategy in sampling-based partitioning process is likely to lead more efficient sample problems. Therefore, we use the domain knowledge based on the Linear Solutions and Binary Solutions in the HNP-MP algorithm in order to get highly efficient sample problems.
B. Calculating the Promise Index, Backtracking, and Stopping Rules

The promising indices for sample problems in the $\ell$ iteration of the HNP-MP algorithm are determined by the lower bounds obtained from the LP relaxations of sample problems. In our proposed hybrid method, we can either use the Dantzig-Wolfe Decomposition method to the LP relaxations first and then implement the Column Generation algorithm to solve them, or use the Simplex method to solve them directly. Both of the methods have their own advantages and disadvantages.

There are many backtracking rule alternatives for the HNP-MP algorithm. The following options for backtracking are considered because they are easy to implement. Backtracking Rule I: Backtrack to the superregion of the current most Promising Region. Backtracking Rule II: Backtrack to the entire feasible region. The difference between these two rules can be thought of in terms of long-term memory. If Rule II is used, then we can move immediately out of that region in a single transition. For Rule I, on the other hand, completely moving out of regions of more depth than one requires more than one transition. Therefore, Rule I has long-term memory, but Rule II does not. In this paper Rule I is used.

In the general Nested Partitions method, the algorithm stops when all integer variables are fixed and the subregion is a singleton. However, we find that stopping partitioning of the current Promising Region in an iteration before a singleton is reached may be advantageous in finding a good approximate solution. Because standard MIP solvers can quickly find a good or even optimal solution within a small region, we adopt the following rule in the HNP-MP algorithm: we check to see if $PFL(\ell)$ becomes sufficiently small in the $\ell$ iteration, and the algorithm stops if $PFL(\ell)$ can be solved using standard MIP solvers in a small amount of time.

V. Computational Results

In the computational tests of all problem instances, we set the total number of samples $N(\ell)$ as 300 in the $\ell$th iteration of the LugNP algorithm. After solving all of the relaxed sample problems, 3 sample problems that have better lower bounds than others are solved by the Relax-and-Fix algorithm, and only the sample problem with the best feasible solution among these 3 problems is considered as the most promising and is used to obtain the next Promising Region and Surrounding Region. In the $\ell$th iteration of the algorithm, the MIP solver time is set as 30 seconds to solve the problem $PFL(\ell)$, the algorithm stops if it can be solved to be optimal. The MIP solver time is set as 50 seconds to solve the subproblem in the Relax-and-Fix algorithm, in which $\alpha$ and $\beta$ are set as 4 and 2, respectively. In Column Generation, the maximum number of iteration is set as 500, and the algorithm stops if the optimal solutions of $J$ subproblems are all less than $10^{-5}$. In the first several iterations of LugNP, the Dantzig-Wolfe Decomposition method and the Column Generation algorithm are used to obtain good original Promising Regions. In the later iterations, the LP Simplex method is used by replacing the Dantzig-Wolfe Decomposition method and the Column Generation algorithm to solve the sample problems in order to save the computational time.

LugNP is coded in GAMS using a solver, CPLEX 10. The tests are done on a computer with the Windows XP Operating System with a 3.0 GHz CPU and 2.0 G RAM. CPU times are given in seconds, the gap is calculated as (the upper bound - the lower bound)/the lower bound. The computational tests on the same instances are also conducted on the methods used in the literature that include the heuristic methods in the LS-LIB and a commercial MIP solver, CPLEX 10. For fair comparison, all approaches are tested in the same computer. Also, the same lower bounds are used to calculate the duality gaps that are listed in the following result tables. The lower bounds used in the paper are generated by the LP relaxations of the FL formulation.

In Subsection V-A and V-B, we report in detail on our computational results with the test instances generated from Trigeiro et al. [12], and compare them with those obtained by heuristic methods in LS-LIB proposed by Pochet et al. [18]. Here we would highlight that LugNP targets on improving upper bound solutions, not lower bound solutions. In Subsection V-C, the computational results are compared among heuristic methods in LS-LIB, CPLEX, and LugNP for hard instances. LS-LIB provides primitives for problem reformulation, cut generation, and heuristics to find good feasible solutions quickly. For the CLST specifically, the most suitable problem reformulation form in the LS-LIB is XFormWWCC(S, Y, D, C, NT, TK, MC), the most suitable cut generation is XCutWWCC(S, Y, D, C, NT), XFormWWCC denotes the single item subproblem reformulation of the Wagner-Whitin relaxation and the CLST with constant production capacity in each period, XCutWWCC represents the cut generation of the Wagner-Whitin relaxation and the CLST with constant production capacity in each period. The parameters in them are explained in the following. S denotes the stock vector, Y denotes the setup vector, D denotes the demand vector, C denotes the constant capacity or the capacity vector, NT denotes the number of time periods, TK is the approximation parameter controlling the size and quality of the reformulation, MC indicates if constraints are added as Model Cuts or are added a priori to the reformulation.

The heuristic methods in the LS-LIB include Relax-and-Fix, Local Branching (LB), Relaxation-Induced-Neighborhood-Search (RINS), and Exchange or Fix-and-Relax, all of these heuristic methods have the forms XHeurRF(CY, SOL, NI, NT, MAXT, FIX, BIN), XHeurLB(CY, SOL, NI, NT, MAXT, PAR), XHeurRINS(CY, SOL, NI, NT, MAXT), and XHeurEXCH(CY, SOL, NI, NT, MAXT, PAR), respectively. In the parameters of these heuristic methods, NI denotes the number of items, NT denotes the number of time periods, CY denotes the constraints indexed over 1..NI, 1..NT defining the Y variables as binary variables, SOL indexed over 1..NI, 1..NT contains as input as initial feasible solution if it is an improvement heuristic, and as output the heuristic solution found, MAXT indicates the maximum time allowed to solve the subproblems. Specifically, in XHeurRF, BIN is the number of binary variables in each subproblem, FIX is the number of additional binary variables fixed in each iteration of Relax-and-Fix, while in others, PAR denotes the number of binary variables relaxed to continuous variables in each iteration.
TABLE I

<table>
<thead>
<tr>
<th></th>
<th>10 Products</th>
<th>20 Products</th>
<th>30 Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utilization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>0.53%</td>
<td>0.64%</td>
<td>0.31%</td>
</tr>
<tr>
<td>Medium</td>
<td>1.24%</td>
<td>1.53%</td>
<td>0.89%</td>
</tr>
<tr>
<td>High</td>
<td>3.78%</td>
<td>8.02%</td>
<td>3.49%</td>
</tr>
<tr>
<td>TBO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>1.31%</td>
<td>1.37%</td>
<td>1.19%</td>
</tr>
<tr>
<td>Medium</td>
<td>1.37%</td>
<td>1.67%</td>
<td>1.23%</td>
</tr>
<tr>
<td>High</td>
<td>2.87%</td>
<td>7.15%</td>
<td>2.29%</td>
</tr>
<tr>
<td>Demand Var</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>1.77%</td>
<td>3.43%</td>
<td>1.44%</td>
</tr>
<tr>
<td>High</td>
<td>1.93%</td>
<td>2.47%</td>
<td>1.69%</td>
</tr>
<tr>
<td>Setup time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>1.81%</td>
<td>2.72%</td>
<td>1.57%</td>
</tr>
<tr>
<td>High</td>
<td>1.89%</td>
<td>4.07%</td>
<td>1.57%</td>
</tr>
<tr>
<td>Average</td>
<td>1.85%</td>
<td>3.40%</td>
<td>1.57%</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th></th>
<th>10 Products</th>
<th>20 Products</th>
<th>30 Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utilization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>6</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>Medium</td>
<td>11</td>
<td>290</td>
<td>52</td>
</tr>
<tr>
<td>High</td>
<td>45</td>
<td>574</td>
<td>155</td>
</tr>
<tr>
<td>TBO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>11</td>
<td>103</td>
<td>42</td>
</tr>
<tr>
<td>Medium</td>
<td>21</td>
<td>159</td>
<td>69</td>
</tr>
<tr>
<td>High</td>
<td>30</td>
<td>648</td>
<td>112</td>
</tr>
<tr>
<td>Demand Var</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>29</td>
<td>410</td>
<td>83</td>
</tr>
<tr>
<td>High</td>
<td>13</td>
<td>196</td>
<td>66</td>
</tr>
<tr>
<td>Setup time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>21</td>
<td>252</td>
<td>79</td>
</tr>
<tr>
<td>High</td>
<td>21</td>
<td>354</td>
<td>70</td>
</tr>
<tr>
<td>Average</td>
<td>21</td>
<td>303</td>
<td>74</td>
</tr>
</tbody>
</table>

Please refer to [18] for a more detailed issue.

A. Comparison of LS-LIB and LugNP for the full factorial experiment

The data set from [12] is randomly generated with some standard distributions proposed by Trigeiro et al.. It includes a total number of 540 instances that are over a time horizon of 20 periods. It is constructed based on a full factorial experiment with five factors: demand variability (medium or high, for each item, average demand is 100 per period), setup time (low or high), per unit capacity utilization (low, medium, or high), setup cost/holding cost ratio (TBO) (low, medium, or high), and number of items (10, 20, or 30). With the specific settings of parameters, the factors are defined as follows: Demand variability (coefficient of variation = 0.35, 0.59), setup time (5.2%, 16.8% of capacity), capacity utilization (75%, 85%, 95%), and TBO (1, 2, 4 periods). A more detailed data generation procedure can be referred from [12]. The computational tests on these data sets are conducted for all heuristic methods in LS-LIB, the results generated by the Relax-and-Fix heuristic method and RINS are better than others. Hence, here we only compare their results with those generated by LugNP. The Relax-and-Fix heuristic method and RINS in LS-LIB are referred to as LS-RF and LS-RINS, respectively. The values of parameters in LS-LIB are defined as follows. TK is set as 3, MC is set as 0 in XFormWWCC. In order to achieve the best results for LS-RF and LS-RINS, MAXT is set as 200 seconds, FIX is set as 2, and BIN is set as 4 in LS-RF, while MAXT is set as 900 seconds in LS-RINS.

In Table I, we present our computational results on duality gaps for all lot sizing problems generated by using different factors, and also compare our results with those obtained by LS-RF and LS-RINS. While in Table II their CPU time is listed. The results show that demand variability and setup time have a minor effect on the gap. If the capacity is more constrained, the problems become more difficult to solve with respect to CPU time. Problems with a higher TBO are also more difficult to solve. The effect on the gap of a low and medium TBO seems only minor, whereas the effect of a high TBO is more apparent. Compared with LS-RF and LS-RINS, LugNP could generate smaller duality gaps on instances with 10 to 20 products, but the computational time is comparatively larger. Furthermore, in terms of instances with 30 products, LugNP could outperform LS-RF and LS-RINS with respect to both duality gaps and computational time.
TABLE III
COMPARISON OF LS-RF, LS-RINS, AND LUGNP

<table>
<thead>
<tr>
<th></th>
<th>LS-RF</th>
<th>LS-RINS</th>
<th>LUGNP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LB</td>
<td>UB</td>
<td>D-Gap</td>
</tr>
<tr>
<td>Tr6-15(G30)</td>
<td>37,103.1</td>
<td>37,996.9</td>
<td>2.41%</td>
</tr>
<tr>
<td>Tr6-30(G62)</td>
<td>60,946.2</td>
<td>61,907</td>
<td>1.58%</td>
</tr>
<tr>
<td>Tr12-15(G53)</td>
<td>73,848.0</td>
<td>74,754</td>
<td>1.23%</td>
</tr>
<tr>
<td>Tr12-30(G69)</td>
<td>130,177.2</td>
<td>130,711</td>
<td>0.41%</td>
</tr>
<tr>
<td>Tr24-15(G57)</td>
<td>136,365.7</td>
<td>136,537</td>
<td>0.13%</td>
</tr>
<tr>
<td>Tr24-30(G72)</td>
<td>287,753.4</td>
<td>288,163</td>
<td>0.14%</td>
</tr>
</tbody>
</table>

*Note that LB denotes lower bounds, UB denotes upper bounds, and D-Gap indicates duality gaps.

TABLE IV
COMPARISON OF LS-RF, CPLEX, AND LUGNP FOR HARD INSTANCES.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TBO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>18.79%</td>
<td>1185</td>
<td>13.66%</td>
<td>3000</td>
<td>8.13%</td>
<td>871</td>
</tr>
<tr>
<td>Medium</td>
<td>13.65%</td>
<td>1164</td>
<td>6.68%</td>
<td>3000</td>
<td>5.78%</td>
<td>884</td>
</tr>
<tr>
<td>High</td>
<td>7.19%</td>
<td>977</td>
<td>4.63%</td>
<td>3000</td>
<td>3.05%</td>
<td>983</td>
</tr>
<tr>
<td>Demand Var</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>15.21%</td>
<td>1512</td>
<td>9.29%</td>
<td>3000</td>
<td>6.06%</td>
<td>1223</td>
</tr>
<tr>
<td>High</td>
<td>11.22%</td>
<td>704</td>
<td>7.37%</td>
<td>3000</td>
<td>5.25%</td>
<td>602</td>
</tr>
<tr>
<td>Setup time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>6.77%</td>
<td>930</td>
<td>6.45%</td>
<td>3000</td>
<td>3.45%</td>
<td>829</td>
</tr>
<tr>
<td>High</td>
<td>19.66%</td>
<td>1287</td>
<td>10.20%</td>
<td>3000</td>
<td>7.86%</td>
<td>996</td>
</tr>
<tr>
<td>Average</td>
<td>13.22%</td>
<td>1108</td>
<td>8.33%</td>
<td>3000</td>
<td>5.66%</td>
<td>912</td>
</tr>
</tbody>
</table>

*Note that -G indicates the duality gap for the corresponding methods, and -T denotes the computational time correspondingly.

B. Comparison of LugNP, LS-RF, and LS-RINS with Other Data Sets

Compared to LS-RF and LS-RINS, LugNP would provide smaller duality gaps. For more comparisons, we take six prominent instances taken from a test set used by [12]. In Table III, we make a comparison on LS-RF and LS-RINS and our proposed LugNP algorithm. Although we cannot make a very robust conclusion based on such a limited comparison, the results of gap improvements show that LugNP could achieve much better solution qualities with a comparatively larger computational time.

C. More Results and Comparisons on Hard Instances

From the computational results presented above, LugNP is able to reduce the duality gaps substantially further when compared to several other heuristic algorithms used in LS-LIB. Here we would like to present computational results on hard problems that have a high capacity utilization. We pick out some hard problems from the full factorial experiment generated by Trigeiro et al. [12]. These hard problems have 30 items and 20 periods with high capacity utilization. Slight changes are made to the settings of parameters for these problems to make them harder, both parameters of setup costs and setup times for all items and periods are multiplied by 4.

VI. CONCLUSION AND FUTURE RESEARCH

In this paper, we propose a new LugNP approach for the capacitated multi-item lot sizing problem with setup times. In the hybrid approach we combine three approaches, Nested Partitions, Column Generation, and Relax-and-Fix together to solve the problem efficiently. To achieve strong lower bounds, we formulate the problem into the classical FL formulation, which is known to provide strong lower bounds. We propose a Dantzig-Wolfe Decomposition reformulation of the FL formulation, and a Column Generation algorithm is implemented to solve its LP relaxation problem, which is able to provide a strong lower bound and a good guidance on Nested Partitions in the hybrid approach. A time-oriented decomposition heuristic, Relax-and-Fix, is also proposed in the hybrid approach to obtain a good initial Promising Region and speed up the algorithm. Computational results show that the LugNP approach provides good results for the CLST. A limited comparison suggests that it is superior to a state-of-the-art system, LS-LIB, and a commercial MIP solver, CPLEX 10. Extensions of this research can focus on both the reformulation and the algorithm. Many extensions of the lot sizing problem have been proposed, such as the overtime allowance case, the backlogging case, the shortage allowance case, and the multilevel production case. It would be interesting to adapt our proposed hybrid approach to all of these extensions.

ACKNOWLEDGMENT

This research was supported in part by the National Science Foundation under grant CMII-00646697, and by the Air Force Office of Scientific Research under grant FA9550-07-1-0390.
REFERENCES


