Accelerate the optimization of large-scale manufacturing planning using game theory

Hui-Ling Zhen1 · Zhenkun Wang2 · Xijun Li3,4 · Qingfu Zhang5,6 · Mingxuan Yuan1 · Jia Zeng1

Received: 30 December 2020 / Accepted: 22 March 2021 / Published online: 9 April 2021 © The Author(s) 2021

Abstract
This paper studies a real-world manufacturing problem, which is modeled as a bi-objective integer programming problem. The variables and constraints involved are usually numerous and dramatically vary according to the manufacturing data. It is very challenging to directly solve such large-scale problems using heuristic algorithms or commercial solvers. Considering that the decision space of such problems is usually sparse and has a block-like structure, we propose to use decomposition methods to accelerate the optimization process. However, the existing decomposition methods require that the problem has strict block structures, which is not suitable for our problem. To deal with problems with such block-like structures, we propose a game theory based decomposition algorithm. This new method can overcome the large-scale issue and guarantee convergence to some extent, as it can narrow down the search space and accelerate the convergence. Extensive experimental results on real-world industrial manufacturing planning problems show that our method is more effective than the world fastest commercial solver Gurobi. The results also indicate that our method is less sensitive to the problem scale comparing with Gurobi.

Keywords Large-scale optimization · Game theory · Decomposition

Introduction
This paper focuses on the solving of large-scale manufacturing planning problems from the industrial applications, which can be modeled as bi-objective Integer Programming (IP) problems. One objective is to maximize the order fill rate (i.e., maximize the satisfaction of requirements). The other is to minimize the total cost (i.e., minimize the operation cost, including inventory cost, production cost and transportation cost). Our goal is to schedule the future manufacture in factories every day. In practical scenarios, the IP problems can easily have over 1 billion decision variables and constraints. It is challenging for commercial solvers such as Gurobi and CPLEX to deal with such large-scale problems efficiently. The larger the problem scale is, the more serious situation becomes.

On one hand, one natural approach to handle the large-scale problems is decomposition. Many decomposition methods such as Benders decomposition [5] and Dantzig-Wolfe decomposition [14] have been well deve-
We propose a novel decomposition algorithm inspired by heuristic constraints, which narrows down the problem scale involved in this paper is much larger than previous ones. A new method based on symmetry-breaking constraints has been utilized for both reformulation and reduction of model in solving large-scale LP and IP [9,10]. Otherwise, there also exists some related research from the field of game [1,4]. They have combined different games with various decomposition algorithms. However, their problems are still far more smaller and simpler than practical ones. To the best of our knowledge, this paper is the first work to use game-based decomposition algorithm to solve billion scale manufacturing planning problem from industrial practice.

Considering the above challenges, we propose a game-based decomposition algorithm and apply it to big manufacturing planning problems. In this algorithm, we first reformulate the IP problem from game perspective. The two objectives are regarded as two players. One is the leader and the other is the follower. Apparently, both two players have relatively small scales, compared to original problem. Then optimizing IP problem is transformed into finding the equilibrium between two players. Different from others, our algorithm is more flexible to deal with non-strict blocked structure.

This paper makes the following major contributions:

1. We propose a novel decomposition algorithm inspired by game, for those non-strict blocked problems.
2. We transform the optimization of IP problem into finding equilibrium between two players, which overcomes the large scale and guarantees the convergence.
3. We construct heuristic constraints, which narrows down the search space, and hence accelerates the convergence.

Experiments are conducted on practical industrial manufacturing planning problems. The results show significant improvements over the best commercial solver Gurobi. It can also be observed that the running time of our algorithm increases much slower than that of Gurobi when problem size increases, which indicates that the proposed algorithm is more friendly to large-scale problems.

| Objective(1a) | z | m | x | y |
|--------------|---|---|---|---|
| Constraint(2b) | √ | √ | √ | √|
| Constraint(2e) | √ | √ | √ | √|

**Model for large-scale manufacturing planning**

In general, the manufacturing planning problem can be modeled as a bi-objective integer programming as below:

\[
\begin{align*}
\text{max} & \quad F(z, m), \\
\text{min} & \quad G(x, y) = c_1^T x + c_2^T y,
\end{align*}
\]

where the \(x, y, z,\) and \(m\) are the decision variables, and they are all positive integers which must satisfy the following constraints:

\[
\begin{align*}
Ax & \leq b, \\
U_1z + U_2m &= \beta, \\
P_1x + P_2y &= \gamma, \\
N_1z + N_2x + N_3y &= \zeta, \\
x &\geq 0, \quad y \geq 0, \quad z \geq 0, \quad m \geq 0.
\end{align*}
\]

Hereby, \(G(x \text{ and } y)\) are both linear, but there are no extra requirements on \(F(z, m)\). Meanwhile, \(c_1, c_2, b, \beta, \gamma, \text{ and } \zeta\) are constant vectors, and \(A, U_1, U_2, P_1, P_2, N_1, N_2,\) and \(N_3\) are constraint matrices. All the constant vectors and constraint matrices are with corresponding dimensions. In practice, both the objectives and constraints have specific meanings.

Figure 1 indicates the mathematical structure of model (1), from which we can find that there exist two main characteristics of the original IP model (1): (1) it has separable objectives. Clearly, one objective is of \(z\) and \(m\), while the other only depends on \(x\) and \(y\). It indicates that we can design an alternative and iterative decomposition method to replace the direct optimization. (2) Its constraint matrix is non-strict blocked. It indicates that the decomposition method must guarantee to converge to the same optima as the original problem. Thus, the conventional decomposition methods cannot
be utilized directly. Considering above, a new decomposition algorithm based on the above properties is proposed in this paper.

**Game-based algorithm**

**Design flow**

**New decomposition from game**

We first decompose the original problem (1) into two subproblems I and II, and both its objective as well as constraints are divided into two parts. It should be noticed that owing to the structure of model, there exists an overlap between the constraints of two subproblems. Assuming that subproblem I could be optimized firstly, and subproblem II is optimized subsequently after the convergence of subproblem I, there exists an apparent alternative and iterative process during the total optimization. Now, we reconsider the two subproblems from game perspective, and then, the two subproblems are treated as two competed players. Let us consider the order from game perspective, and then, the two subproblems are total optimization. Now, we reconsider the two subproblems exist an apparent alternative and iterative process during the following steps. We first give the approximate dual form of problem (4) as below:

1) Solve problem (3) with optimal solutions \((z^*, m^*)\). From game perspective, that is the leader moves first.
2) Solve problem (4) after substituting \((z^*, m^*)\) with optimal solutions \((x^*, s^*, v^*)\). It indicates that the follower finds a best response to the decision of leader.
3) Solve problem (3) again after fixing \((x^*, s^*, v^*)\) and an additional constraint given by the follower’s response, with new solutions \((z^*_1, m^*_1)\). It denotes the leader’s adjustment according to the response of follower.
4) Problem (4) is solved again with \((z^*_1, m^*_1)\) substituted. That implies the follower modifies response according to leader’s latest strategy.

The two players will stop when they cannot find better strategies in the next loop. It should be mentioned that after the follower’s first response, we must add an additional constraint to the leader and such constraint does not exist in the original problem. In practice, we usually obtain the additional constraint in the following steps. We first give the approximate dual form of problem (4) as below:

1) Solve problem (3) with optimal solutions \((z^*, m^*)\). From game perspective, that is the leader moves first.
2) Solve problem (4) after substituting \((z^*, m^*)\) with optimal solutions \((x^*, s^*, v^*)\). It indicates that the follower finds a best response to the decision of leader.
3) Solve problem (3) again after fixing \((x^*, s^*, v^*)\) and an additional constraint given by the follower’s response, with new solutions \((z^*_1, m^*_1)\). It denotes the leader’s adjustment according to the response of follower.
4) Problem (4) is solved again with \((z^*_1, m^*_1)\) substituted. That implies the follower modifies response according to leader’s latest strategy.

The two players will stop when they cannot find better strategies in the next loop. It should be mentioned that after the follower’s first response, we must add an additional constraint to the leader and such constraint does not exist in the original problem. In practice, we usually obtain the additional constraint in the following steps. We first give the approximate dual form of problem (4) as below:

\[
\begin{align*}
\text{max} \quad & b^T \tilde{x}, \quad \text{max} \quad \sigma \tilde{x}, \quad \text{max} \quad \sigma \tilde{y} \\
\text{s.t.,} \quad & A^T \tilde{x} \geq c_1, \quad P_1^T \tilde{x} = c_1, \quad N_1^T \tilde{x} = c_1 \\
& \tilde{y} \geq 0, \quad \tilde{x} \geq 0, \quad \tilde{y} \geq 0.
\end{align*}
\]
Algorithm 1: Game-based decomposition algorithm

1: **Input**: raw optimization problem (1), tolerance value $\epsilon$, iteration counter $\gamma = 1$.
2: Construct two players, i.e., subproblem (3) and (4).
3: **Update rule**:
4: while Strategy difference $> \epsilon$ do
5: Optimize problem (3)
6: Optimize problem (4)
7: Optimize approximate dual (5)
8: if $\gamma > 1$ then
9: Update Lagrangian dual parameter in subgradient direction
10: end if
11: Add heuristic constraints (6)
12: Optimize problem (3)
13: Optimize problem (4)
14: $\gamma = \gamma + 1$
15: end while
16: **Output**: Optimal solutions.

Illustrations on algorithmic advantage

Figure 2 shows the overview of our game-based algorithm. Notice that the additional constraints, which are given by the response of follower, are displayed with a different color in this figure.

Back to other given alternative and iterative algorithms [5,15] in solving large-scale optimization problem, the new proposed game-based decomposition method has the following different points:

(a) Different from the given decomposition algorithm, the order of different subproblems does matter during the optimization. Recall that the two subproblems are leader and follower, respectively. It should be mentioned that for leader, its solution matters for the total iterative process, since it provides the initial point for optimizing the follower; while for follower, its response is added to the leader and it reflects the convergence efficiency. In practice, we prefer to choose the subproblem that is relatively easy to be solved as the leader.

(b) Different from the given alternative and iterative algorithm, we must add the response of follower as the additional constraints during optimization. Figure 3 shows the effect of additional constraints. The abscissa is $F(z, m)$ and ordinate is $G(x, y)$. They change along the given directions. Based on the settings, higher quality solutions mean larger $F(z, m)$ but smaller $G(x, y)$. Assume that Gurobi optimize original problem (1) in a weight-sum way. The dotted curve is obtained by Gurobi with different weights adjusted. Although the curve is similar as the Pareto front, no one can guarantee its Pareto optima. The dashed lines represent the results of game-based decomposition. It is clear that as $F(z, m)$ increases and $G(x, y)$ decreases, the game-based decomposition can converge to a solution, whose quality is no lower than the solutions given by Gurobi directly.

(c) Different from the given decomposition algorithms which aim at finding the Nash equilibrium between sub-problems, our algorithm focuses on the Stackelberg equilibrium. According to the game theory [3,11], the Stackelberg equilibrium between the leader and follower guarantees the convergence and solution quality. More discussions are given in Sect. 4.

Theoretical guarantee

Preliminaries of game

Figure 4 shows the difference between Nash and Stackelberg equilibrium.
Results of the game-based algorithm and heuristic constraints. Abscissa is $F(z, m)$, while ordinate is $G(x, y)$. The dotted curve denotes the solutions given by Gurobi in a weight-sum way directly. The dashed and solid lines are from our algorithm with and without heuristic constraints, respectively. It shows the convergence of game-based decomposition. Moreover, heuristic constraints not only accelerate the convergence but also lead to higher quality solutions.

Graphical illustration for the flow of game-based decomposition

(i) if the leader $A$ chooses the strategy $a$ first, then the optimal solutions of $F_B$ can be expressed as $f(a)$ which goes across its global optimal strategy $Q$, and Stackelberg equilibrium $(a_S, b_S)$ denotes the tangent of $F_A$ and $f(a)$. Obviously, the follower will choose the strategy which is the most favorable for leader.

(ii) if players $A$ and $B$ do not share information before decision, $(a_N, b_N)$ is a Nash equilibrium, since $F_A$ has a horizontal tangent at this point, while $F_B$ has a vertical tangent. It denotes that one cannot increase his payoff by single-mindedly changing his own strategy, as long as the other sticks to the Nash equilibrium.

Back to our problem, the leader (3) (i.e., $A$ in Fig. 4) adopts the strategy $a$, and the follower (4) (i.e., $B$ in Fig. 4) requires to maximize $F_B(a, b)$ and chooses a best reply $b^* = f(a)$, the goal of leader is now to maximize $F_A(a, f(a))$. Assuming that player $A$ servers as the leader and announces his strategy in advance, then player $B$ makes his decisions accordingly. In Pareto optima, one cannot increase its own payoff strictly without decreasing the payoff of the other.

**Theoretical discussions**

Theorem 1 first guarantees the solution quality of game-based decomposition under strict mathematical sense. It denotes that the game-based decomposition can converge to the optimal solution of original problem under certain mathematical assumptions, where the two players cannot find better strategy in the next loop.

**Theorem 1** When $\exists B_1, B_2 \ s.t. B_1u_1 + B_2u_2 = 0$, the leader and follower can converge to Stackelberg equilibrium, which is the optima of original problem.

**Proof** Supposing that the two players are differentiable, we label the follower via $u_1$, while $u_2$ refers to the leader. Due to that, they are differentiable, and the cost functions for $u_1$ and $u_2$, respectively, can be given as:

\[
J_1(x, u_1, u_2) = \frac{1}{2} \int_0^\infty r_1(x, u_1, u_2)dt,
\]

\[
J_2(x, u_1, u_2) = \frac{1}{2} \int_0^\infty r_2(x, u_1, u_2)dt.
\]
where
\[ r_1(x, u_1, u_2) = x^T Q_1 x + u_1^T R_{11} u_1 - u_1^T R_{12} u_2, \]
\[ r_2(x, u_1, u_2) = x^T Q_2 x - u_1^T R_{21} u_1 + u_2^T R_{22} u_2, \]
and \( Q_j \) and \( R_{jk} \) are both positive definite and symmetric. It is clear that the cost function of every player hopes to optimize its own function and minimize the partner’s. Owing to the different levels of \( u_1 \) and \( u_2 \), we can first obtain the optimal solutions of follower as:
\[ J^*_f(x, u_1, u_2) = \min_{u_1} \frac{1}{2} \int_{t}^{\infty} r_1(x, u_1, u_2) dt, \]
and the follower’s Hamiltonian is:
\[ H_{u_1} = r_1(x, u_1, u_2) + (\nabla J_1)^T (A x + B_1 u_1 + B_2 u_2). \]
Thus, the optimal controller can be obtained as \( \frac{\partial H_{u_1}}{\partial u_1} = 0 \), that is:
\[ u_1^* = - \frac{1}{2} (R_{11})^{-1} B_1^T \nabla J_1. \]
Since the follower’s optimal solutions can affect the optimization of leader, the optimal solutions of leader can be expressed as:
\[ J^*_l(x, J^*_f, u_2) = \min_{u_2} \frac{1}{2} \int_{t}^{\infty} r_2(x, J^*_f, u_2) dt, \]
with \( r_2(x, J^*_f, u_2) = r_2(x, u_1, u_2) \bigg|_{u_1=J^*_f} \). Utilizing the gradient of follower’s Hamiltonian:
\[ \frac{\partial H_{u_1}}{\partial x} = (Q_1 + Q_1^T) x + A^T \nabla J_1, \]
we can obtain the optimal cost function of leader:
\[ J^*_l(x, J^*_f, u_2) = \min_{u_2} \frac{1}{2} \int_{t}^{\infty} \left[ x^T Q_2 x + (u_1^*)^T R_{21} u_1^* + u_2^T R_{22} u_2 + \lambda \frac{\partial H_{u_1}}{\partial x} \right] dt. \]
Set \( r_2(x, J^*_f, u_2) = x^T Q_2 x + (u_1^*)^T R_{21} u_1^* + u_2^T R_{22} u_2 + \gamma \frac{\partial H_{u_1}}{\partial x} \), we have the leader’s Hamiltonian as:
\[ H_{u_2} = \gamma \left[ (Q_1 + Q_1^T) x + A^T \nabla J_1 \right] + r_2(x, J^*_f, u_2) + (\nabla J_2)^T (A x + B_1 J^*_f + B_2 u_2), \]
with \( \gamma \) as the real vector\(^1 \). To get the optimal controller, we have \( \frac{\partial H_{u_2}}{\partial u_2} = 0 \), which leads to:
\[ u_2^* = - \frac{1}{2} (R_{22})^{-1} B_2^T \nabla J_2. \]
Obviously, we also have:
\[ \nabla u H_{stack} = \left( \frac{\partial H_{u_1}}{\partial u_1}, \frac{\partial H_{u_2}}{\partial u_2} \right), \]
and gradients of Hamiltonians for leader and follower also satisfy:
\[ (\xi(u_1, u_2), (\nabla u H_{stack})^T) = 0, \]
with \( \xi(u_1, u_2) = (2R_{11} u_1, 2R_{22} u_2) \) and \( B_1 u_1 + B_2 u_2 = 0 \), i.e., the gradients on follower’s and leader’s Hamiltonians converge to the equilibrium from game perspective.

Considering the complicated circumstance in practice, Theorem 2 then guarantees the solution quality with a compact upper bound.

**Theorem 2** The proposed game-based decomposition can approximate the optima of original problem, since optimal solutions of leader can be bounded by the dual form and optimal solutions of follower.

**Proof**. To prove Theorem 2, we first construct a toy model:
\[
\min c^T x + d^T y + h^T z, \\
s.t., \quad Ax \leq b, \\
Mx + Ny + Uz \geq v, \tag{13}
\]
to show why we can add a constraint to leader with the optimal solutions and dual form of follower. Hereby, \( x, y, \) and \( z \) are the decision variables, \( c \) and \( d \) are constant vectors, and \( A, M, \) and \( N \) are constant matrices with corresponding dimensions. Now, as the settings in our game-based decomposition algorithm, problem (13) is decomposed into two problems: the leader:
\[
\min c^T x + \alpha, \\
s.t., \quad Ax \leq b, \\
\alpha \geq \alpha_{down}, \tag{14}
\]
and the follower:
\[
\min d^T y + h^T z, \\
s.t., \quad Ny + Uz \leq v - M\hat{x}, \tag{15}
\]
\(^1\) Considering the optimization process, it serves as a Lagrangian multiplier.
where $\hat{\mathbf{x}}$ is given by the optimal solutions of leader (14). Different from [5], there exist two variables in the follower. A naive method is to use the nested form to give the dual form, which only has one decision variable in one loop, as nested Benders decomposition [13]. However, we think this method costs too much time in large-scale problem. Next, we give the dual form of follower (15) in an approximate way:

$$
\min \left[ \mathbf{v} - \mathbf{M}^T \hat{\mathbf{y}} \right] (\mathbf{y} + \hat{\mathbf{z}}), \\
s.t., N^T \mathbf{y} \leq \mathbf{d}, \\
\mathbf{U}^T \hat{\mathbf{z}} \leq \mathbf{h}, \\
\hat{\mathbf{y}} \geq 0,
$$

(16)

where $\mathbf{y}$ and $\hat{\mathbf{z}}$ are the dual variables corresponding with $\mathbf{y}$ and $\mathbf{z}$, respectively. Then, for the original problem (13), we have:

$$
\min \mathbf{x} \left\{ \mathbf{c}^T \mathbf{x} + \min_{\mathbf{y}} \left\{ \mathbf{d}^T \mathbf{y} + \mathbf{h}^T \mathbf{z} \left| N^T \mathbf{y} \leq \mathbf{d}, \mathbf{U}^T \hat{\mathbf{z}} \leq \mathbf{h} \right. \right\} \\
= \min \mathbf{x} \left\{ \mathbf{c}^T \mathbf{x} + \max_{\mathbf{y}} \left\{ \left[ \mathbf{v} - \mathbf{M} \hat{\mathbf{y}} \right]^T \mathbf{y} \right\} \\
\leq \min \mathbf{x} \left\{ \mathbf{c}^T \mathbf{x} + \max_{\mathbf{y}} \left\{ \left[ \mathbf{v} - \mathbf{M} \hat{\mathbf{y}} \right]^T \mathbf{y}, \left[ \mathbf{v} - \mathbf{M} \hat{\mathbf{y}} \right]^T \mathbf{z} \right\} \right\} \\
\leq \min \mathbf{x} \left\{ \mathbf{c}^T \mathbf{x} + \max_{\mathbf{y}} \left\{ \left[ \mathbf{v} - \mathbf{M} \hat{\mathbf{y}} \right]^T \mathbf{y}, \left[ \mathbf{v} - \mathbf{M} \hat{\mathbf{y}} \right]^T \mathbf{z} \right\} \right\}
$$

where $\mathbf{y}^*$ and $\mathbf{z}^*$ are the optimal solutions of corresponding dual problems. Thus, Theorem 2 is proved. 

### Experimental evaluation

Now, we take a practical manufacturing planning task from our company as an example, to show the deployment details for the proposed game-based decomposition algorithm. In this example, $i \in I = I_P \cup I_A$ denotes all kinds of products; $p$ and $p'$ are both for plant; and $t \in [0, T]$ refers to the time period. $I_P$ and $I_A$ represent the sets of semi-finished and finished products, respectively. Other related notations are presented in Table 1.

| Variables | Notations |
|-----------|-----------|
| $g_{1}(x)$ | Supply quantity of item $i$ of plant $p$ |
| $g_{2}(s)$ | Transportation quantity of item $i$ of plant $p$ at time $t$ |
| $g_{3}(x, s, v)$ | Planning production quantity of item $i$ of plant $p$ at time $t$ |
| $m_{i,t}$ | Delay quantity of item $i$ at time $t$ |
| $r_{i,p,t}$ | Replacement quantity of item $i$ of plant $p$ at time $t$ |
| $v_{i,p,t}$ | Inventory constraint of item $i$ of plant $p$ at time $t$ |
| $i_{i,t}$ | Inbound constraint of item $i$ of plant $p$ at time $t$ |
| $o_{i,p,t}$ | Outbound constraint of item $i$ of plant $p$ at time $t$ |
| $D_{i,t}$ | Aggregate demand at time $t$ |
| $F_{i,t}$ | Predicted demand at time $t$ |
| $p_{i}$ | Plants set of item $i$ |
| $p'_{i}$ | Planning production quantity of item $i$ |
| $LT_{p}$ | Lead time |
| $PC_{i,p}$ | Processing cost of item $i$ of plant $p$ |
| $PT_{i,p}$ | Processing time of item $i$ of plant $p$ |
| $TC_{i,p}'$ | Transportation cost of item $i$ from plant $p$ to plant $p'$ |
| $MC_{i}$ | Material cost of item $i$ |
| $V_{i,p}$ | Initial inventory of item $i$ of plant $p$ |
| $B_{i,p,t}$ | Cost for unit quantity |
| $U_{i,p,c}$ | Unit occupancy of item $i$ of plant $p$ |
| $CAP_{i,p,t}$ | Upper limit for production |
| $MLS_{i,p,t}$ | Minimal production quantity of item $i$ of plant $p$ at time $t$ |
| $PAIR_{i,p,t}$ | Processing ratio of item $i$ from plant $p$ to plant $p'$ |
| $R_{i,p,t}$ | Total replacement quantity of item $i$ of plant $p$ at time $t$ |
| $\Delta_{1}, \Delta_{2}$ | Delay owing to holidays |

where $g_{1}(x)$, $g_{2}(s)$, and $g_{3}(x, s, v)$ refer to the manufacturing, transportation, and holding costs, respectively:

$$
\begin{align*}
g_{1}(x) &= \sum_{i \in I_P} \sum_{p \in P} PC_{i,p} \cdot \left( \sum_{t} PM_{i} \cdot x_{i,t} \right) \\
g_{2}(s) &= \sum_{i \in I} \sum_{p \in P} \left( \sum_{p' \in \{P \setminus P'\}} TC_{i,p}' \cdot \left( \sum_{t} x_{i,p',t} \right) \right) \\
g_{3}(x, s, v) &= \sum_{i \in I} \sum_{p \in P} HC_{i,p} \cdot \left( \sum_{t} v_{i,t} \right) + \sum_{i \in I} \sum_{p \in P} HC_{i,p} \cdot \left( LT_{p} \cdot \sum_{t} x_{i,p',t} \right) + \sum_{i \in I} \sum_{p \in P} HC_{i,p} \cdot \left( PT_{i,p} \cdot \sum_{t} PM_{i} \cdot x_{i,t} \right).
\end{align*}
$$

### Model

In this subsection, we ignore the problem details and focus on demonstrating the core problem structure. Hereby, the two objectives are embodied as the order fillrate $F(z, m)$ and total cost $G(x, s, v)$, respectively:

$$
\max F(z, m) = \sum_{i \in I, t \in T} \frac{z_{i,t} - m_{i,t}}{D_{i,t} + F_{i,t}}
$$

(17)

$$
\min \ G(x, s, v) = g_{1}(x) + g_{2}(s) + g_{3}(x, s, v),
$$

(18)
Meanwhile, there are many constraints to be considered. Among them, three are corresponding with constraint (2a):

\[
\sum_{i \in I_i} U_{i,p,c} \cdot (PM_i \cdot x_{i,p,t}) \leq CAP_{p,t,s,c}
\]

\[\text{(19)}\]

\[
\text{PM}_i \cdot x_{i',p,t} = PAIR_{i',p,t} \cdot (PM_i \cdot x_{i,p,t}), \text{ } i' \neq i
\]

\[\text{(20)}\]

\[
\text{PM}_i \cdot x_{i,p,t} \in \{0, MLS_{i,p,t}, MLS_{i,p,t} + 1, \ldots\}
\]

\[\text{(21)}\]

Constraint (19) is the limitation of production capacity. Constraint (20) is for the lot size of each productivity, which means that the goods must be produced in pair. Constraint (21) denotes the minimal production of every plant. The limitation on delay corresponds with constraint (2b):

\[
m_{i,t} = \begin{cases} M_{i,t} & t = 0 \\ m_{i,t-1} + D_{i,t} - \sum_{p \in P_i} z_{i,p,t}, & \text{otherwise.} \end{cases}
\]

\[\text{(22)}\]

Limitations on inventory and inbound correspond with constraint (2c):

\[
v_{i,p,t} = \begin{cases} V_{i,p} & t = 0 \\ v_{i,p,t-1} + i_{n, p,t} - i_{t, p,t}, & \text{otherwise.} \end{cases}
\]

\[\text{(23)}\]

\[
i_{n, p,t} = \sum_{p' \in P_i} s_{i',p,t-LT_{i,p} - \Delta_1} \]

\[\text{PM}_i \cdot x_{i,p,t} - PT_{i,p} - \Delta_2 + PO_{i,p,t}
\]

\[\text{(24)}\]

Limitations on outbound and replacements relate to constraint (2d):

\[
\text{out}_{i,p,t} = \sum_{p' \in P_i} s_{i',p,t} + \left[ \sum_{i' \in I_{i',p}} \left( B_{i',p,t} \cdot PM_i \cdot x_{i',p,t} \right) \right]

\[+ z_{i,p,t} - \sum_{i' \in I_{i',p}} r_{i',p,t} \right] + \sum_{i' \in I_{i',p}} r_{i',p,t}
\]

\[\text{(25)}\]

\[
\sum_{i' \in I_{i',p}} \tilde{x}_{i',p} + z_{i,p,t} - \sum_{i' \in I_{i',p}} r_{i',p,t} \leq 0,
\]

\[\text{(26)}\]

with \(\tilde{x}_{i',p} = B_{i',p,t} \cdot (PM_i \cdot x_{i',p,t})\). Note that constraint (26) denotes that item \(i'\) should first satisfy its own father node, and then serves for other leaf nodes as the replacement. All the related variables are positive.

**Implementation details**

According to Algorithm 1, the implementation details for such example is given below:

- **Input**: Manufacturing planning problem.
- **Output**: Optimal solutions.

**Step 0**: Initialization:

- **Step 0.1**: Tolerance value \(\epsilon\), iteration counter \(\gamma = 1\), and hyperparameter \(a_{\text{initial}}\).
- **Step 0.2**: We reformulate this problem from game perspective:
  
  Leader: max \(F(z, m)\), with constraints (22), (23) (24) and (25). Follower: min \(G(x, s, v)\), with all constraints apart from (22).

**Step 1**: Solve the modified leader as below:

\[\max F(z, m) + \alpha,\]

s.t. Constraint delay, inventory, outbound and inbound

\[\alpha \geq a_{\text{initial}}, \quad 0 \leq x \leq x_{up},\]

Fixed the solutions \(z = z^{(\gamma)}\) and \(m = m^{(\gamma)}\).

**Step 2**: Solve follower \(G(x, s, v)\) with fixed \(z = z^{(\gamma)}\) and \(m = m^{(\gamma)}\) and obtain \(x = x^{(\gamma)}, s = s^{(\gamma)}, v = v^{(\gamma)}\), \(r = r^{(\gamma)}, in = in^{(\gamma)}\) and \(out = out^{(\gamma)}\).

**Step 3**: Convergence check.

- **Step 3.1**: Compute an upper bound (UB) as below:

\[\Omega^{(\gamma)}_{\text{upper}} = \beta_1^{(\gamma)}[g_1(x^{(\gamma)}) + g_2(s^{(\gamma)}) + g_3(x^{(\gamma)}, s^{(\gamma)}, v^{(\gamma)})]

\[+ \beta_2^{(\gamma)} \sum_{p \in P_i} \left( z_{i,p,t} - m_{i,t-1}^{(\gamma)} \right) D_{i,t} + F_{i,t}\]

where \(\beta_1^{(\gamma)}\) and \(\beta_2^{(\gamma)}\) are two hyperparameters.

- **Step 3.2**: Compute a lower bound (LB) as below:

\[\Omega^{(\gamma)}_{\text{lower}} = \beta_1^{(\gamma)}[a^{(\gamma)} + \beta_2^{(\gamma)} \sum_{p \in P_i} \left( z_{i,p,t} - m_{i,t-1}^{(\gamma)} \right) D_{i,t} + F_{i,t}\]

**Step 3.3**: If \(\Omega^{(\gamma)}_{\text{upper}} - \Omega^{(\gamma)}_{\text{lower}} < \epsilon\), stop, the optimal solutions are obtained. Otherwise, the algorithm continues to the next step.

**Step 4**: Calculate dual problems. We obtain the approximate dual form (I) of follower \(G(x, s, v)\), according to (5).

**Step 5**: Update Leader, i.e., resolve Leader (27) with heuristic constraints,

\[\max F(z, m) = \sum_{i \in I, t \in T} \frac{\sum_{p \in P_i} (z_{i,p,t} - m_{i,t-1}^{(\gamma)})}{D_{i,t} + F_{i,t}} + \alpha,\]

s.t. \(m_{i,t} = \begin{cases} M_{i,t} & t = 0 \\ m_{i,t-1} + D_{i,t} - \sum_{p \in P_i} z_{i,p,t}, & \text{otherwise} \end{cases}\)

\[\alpha \geq a_{\text{initial}}, \quad 0 \leq x \leq x_{up},\]

\[g_1(x^{(\gamma)}) + g_2(s^{(\gamma)}) + g_3(x^{(\gamma)}, s^{(\gamma)}, v^{(\gamma)})\]
tion capacity and large demand. 3
marketing and employment situation, with limited produc-
regular quantities. Datasets I-2 and II-2 are from irresistible
and production capacity of related factories are both with
normal manufacturing instances, where the order demand
on a variety of problems. Datasets I-1, II-1, and III are from
problems with millions of variables, to test its effectiveness
ables in the problem. Here, we also evaluate our algorithm on
relaxation of IP problems 2.

Table 2  Experimental datasets

| Dataset no. | Item   | Plant | Time span | Scale |
|-------------|--------|-------|-----------|-------|
| Dataset I-1 | 996    | 11    | 30        | Million |
| Dataset I-2 | 996    | 11    | 30        | Million |
| Dataset II-1| 4409   | 20    | 30        | 10 million |
| Dataset II-2| 4409   | 20    | 30        | 10 million |
| Dataset III | 66110  | 68    | 30        | Billion |

\[ + \sum_{i \in I} \lambda_{1,i}^{(k)} (z_i - z_i^{(k)}) + \sum_{i \in I} \lambda_{2,i}^{(k)} (m_i - m_i^{(k)}) \leq \alpha, \]

where \( \lambda_{1,i}^{(k)} \) and \( \lambda_{2,i}^{(k)} \) (\( k = 1, \ldots, \gamma \)) are Lagrangian dual parameters, and they change along the corresponding subgradient direction in each last iteration.

- **Step 6**: Update \( \gamma = \gamma + 1 \), and back to Step 1.

It is worth mentioning that in practice, we always choose a relative easy player as the leader.

**Numerical results**

In this section, we evaluate the game-based decomposition algorithm in industrial applications. All baselines are given by Gurobi 8.1 [2,16]. To simplify, we only evaluate the linear relaxation of IP problems 2.

**Datasets**

The experimental datasets are from real-world manufacturing planning applications, and their statistics are shown in Table 2. Here, column Item denotes the number of products to be manufactured. Column Scale lists the number of variables in the problem. Here, we also evaluate our algorithm on problems with millions of variables, to test its effectiveness on a variety of problems. Datasets I-1, II-1, and III are from normal manufacturing instances, where the order demand and production capacity of related factories are both with regular quantities. Datasets I-2 and II-2 are from irresistible marketing and employment situation, with limited production capacity and large demand.\(^3\)

**Solution quality evaluation**

We first evaluate the algorithm on solution quality, as shown in Figs. 5, 6, 7, 8, and 9. Horizontal axis refers to the order fillrate, and vertical axis refers to the total cost. Clearly, our game approach can lead to higher quality solutions, with higher order fillrate and smaller cost than that obtained from Gurobi. It is easy to understand that higher fillrate always costs more. There is no standard trade-off between fillrate and cost. In this sense, the higher quality solutions denote higher fillrate but with smaller cost.

Obviously, in most cases, our algorithm can dominate Gurobi on solution quality. In additions, we also find that in some special cases, our algorithm leads to higher fillrate with more cost, e.g., 11/30-12/29 and 01/02-02/01 in Dataset I-1, 11/21-12/20 and 11/28-12/27 in Dataset II-1. We consider the latter case as our additional advantage over Gurobi, since in some specific manufacturing planning tasks, maximizing the order fillrate is more important than minimizing the cost. Therefore, we can claim that the proposed algorithm outperforms the best commercial solver Gurobi on solution quality.

**Efficiency evaluation**

Figure 10 shows the comparison of computational efficiency. The upper part is from our algorithm, and the lower part is from Gurobi. Clearly, when the problem is small, Gurobi is much more efficient than ours. However, with the increasing of problem scale, our algorithm’s efficiency improves sharply comparing with Gurobi. Figure 11 shows the comparison on time increment when problem scale increases. We can observe that as the running time of Gurobi has 1682-fold increment, while the running time of our algorithm only has 517-fold increment. Our algorithm’s efficiency drops much slower than Gurobi’s when the problem scale increases.

It is because the computation of dual problems and construction of heuristic bounds would cost a certain amount of time. It is very expensive in a relatively small-scale problem. However, with scale increasing, the percentage of time consumed by calculating dual forms and constructing heuristic constraints drops relatively. Therefore, our algorithm is less sensitive to the problem scale comparing with Gurobi.

**Conclusions**

Big industrial manufacturing planning problems bring great challenges to commercial solvers. In practice, these problems have up to billions decision variables and constraints. This paper has proposed a game-based decomposition algorithm to deal with these big problems. Experiments on industrial datasets have shown our improvements on solution quality and robustness. Furthermore, it can be observed that our algo-

\(^2\) Based on theory of IP, people can obtain the IP solutions based on its relaxation.

\(^3\) We take an example to show their difference: in normal cases, the order needs a product \( i \) with the quantity 1800, while production capacity of all related factories can reach 9e11. However, in unnormal cases, the order needs a product \( i \) with the quantity 1e6, but the production capacity of every related factory is less than 200.

2 Based on theory of IP, people can obtain the IP solutions based on its relaxation.

3 We take an example to show their difference: in normal cases, the order needs a product \( i \) with the quantity 1800, while production capacity of all related factories can reach 9e11. However, in unnormal cases, the order needs a product \( i \) with the quantity 1e6, but the production capacity of every related factory is less than 200.
Different from other decomposition algorithms, our algorithm can deal with non-strict blocked problems. Our major algorithm's efficiency decreases much slower than Gurobi's as problem scale increases.
Contributions include: (1) A new decomposition algorithm inspired by games, which is different from previous works and can deal with non-strict blocked problems; (2) A new optimization process, which overcomes the large scale and converges to a solution; (3) A construction of heuristic constraints, which can narrow down the search space and accelerate the convergence.

To the best of our knowledge, this is the first work to apply game-based decomposition algorithms for billion-scale industrial manufacturing planning problems. The algorithm demonstrates significant improvement over state-of-the-art commercial solver Gurobi on solution quality, robustness, and extensibility to large-scale problems. In the future, we will continue to study the efficient algorithms for industrial large-scale tasks in supply chain management and scheduling.

Acknowledgements This work was supported by the National Key Research and Development Project, Ministry of Science and Technology, China (Grant no. 2018AAA0101301), the National Natural Science Foundation of China (Grant no. 61876163), and the ANR/RGC Joint Research Scheme sponsored by the Research Grants Council of the Hong Kong Special Administrative Region, China and France National Research Agency (Project no. A-CityU101/16).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Aazami A, Saidi-Mehrabad M (2018) Benders decomposition algorithm for robust aggregate production planning considering pricing decisions in competitive environment: a case study. Sci Iran
2. Bixby RE (2012) A brief history of linear and mixed-integer programming computation. Doc Math 20:107–121
3. Bloem M, Alpcan T, Başar T (2007) A Stackelberg game for power control and channel allocation in cognitive radio networks. In: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools. ICST (Institute for Computer Sciences, Social-Informatics), p 4
4. Casorrán-Amilburu C (2017) Formulations and algorithms for general and security Stackelberg games. Ph.D. thesis
5. Conejo AJ, Castillo E, Mínguez R, García- Bertrand R (2006) Decomposition techniques in mathematical programming: engineering and science applications. Springer, Berlin
6. Fischetti M, Ljubić I, Sinnl M (2016) Redesigning Bender's decomposition for large-scale facility location. Manag Sci 63(7):2146–2162
7. Grover A, Markov T, Attia P, Jin N, Perkins N, Cheong B, Chen M, Yang Z, Harris S, Chueh W, et al (2018) Best arm identification in multi-armed bandits with delayed feedback. arXiv:1803.10937 (arXiv preprint)
8. Haase K (2012) Lotsizing and scheduling for production planning, vol 408. Springer, Berlin
9. Jans R (2009) Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. INFORMS J Comput 21(1):123–136
10. Lima RM, Novais AQ (2016) Symmetry breaking in MILP formulations for unit commitment problems. Comput Chem Eng 85:162–176
11. Myerson RB (2013) Game theory. Harvard University Press, Harvard
12. Pochet Y, Wolsey LA (2006) Production planning by mixed integer programming. Springer, Berlin
13. Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The benders decomposition algorithm: a literature review. Eur J Oper Res 259(3):801–817

14. Vanderbeck F (2000) On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Oper Res 48(1):111–128

15. Velasquez J, Khakifirooz M, Fathi M (2019) Large scale optimization in supply chains and smart manufacturing-theory and applications

16. Yu W, Chavez R, Jacobs MA, Feng M (2018) Data-driven supply chain capabilities and performance: a resource-based view. Transport Res Part E Logist Transport Rev 114:371–385

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.