Complex & Intelligent Systems (2022) 8:2719–2730
https://doi.org/10.1007/s40747-021-00352-7

ORIGINAL ARTICLE

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

Hui-Ling Zhen[1] · Zhenkun Wang[2] · Xijun Li[3,4] · Qingfu Zhang[5,6] · Mingxuan Yuan[1] · Jia Zeng[1]

Received: 30 December 2020 / Accepted: 22 March 2021 / Published online: 9 April 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 fillrate (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*C*Wolfe decomposition [14] have been well devel-

✉ Zhenkun Wang
wangzk3@sustech.edu.cn

Hui-Ling Zhen
zhen.huiling@huawei.com

Xijun Li
xijun.li@huawei.com

Qingfu Zhang
qingfu.zhang@cityu.edu.hk

Mingxuan Yuan
Yuan.Mingxuan@huawei.com

Jia Zeng
Zeng.Jia@huawei.com

[1] Huawei Noah's Ark Lab, Hong Kong, China

[2] School of System Design and Intelligent Manufacturing, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

[3] Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China, Hefei, China

[4] Huawei Noah's Ark Lab, Shenzhen 518173, China

[5] Department of Computer Science, City University of Hong Kong, Hong Kong, China

[6] Shenzhen Research Institute, City University of Hong Kong, Shenzhen 518057, China

oped. There are also many tricks to accelerate the decomposition methods [5,8,12]. However, these decomposition methods require that the problems satisfy the strict blocked structures. Most practical problems are too complicated to meet the requirement.

On the other hand, there are many heuristic methods have been proposed. The most related work is [6]. It has replaced part of integer variables with heuristic constraints. Experimental results on facility location tasks show its advantage over CPLEX. In this paper, we adopt the similar idea of heuristic constraints. Another related work is [7], in which data-driven algorithms have been proposed to boost solvers. Nevertheless, these approaches are to solve relatively small problems. The problem scale involved in this paper is much larger than previous ones. A new method based on symmetry-breaking constraints have 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 firstly 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.

	z	m	x	y
Objective(1a)	√	√		
Constraint(2b)	√	√		
Constraint(2d)	√		√	√
Objective(1b)	Intersectional parts		√	√
Constraint(2a)			√	
Constraint(2c)			√	√

Fig. 1 Structure of constraint matrix. Clearly, the constraint matrix is not rigorously blocked. Rank of constraints is adjusted, to show the structure clearly

Model for large-scale manufacturing planning

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

$$\max \ F(\mathbf{z}, \mathbf{m}), \tag{1a}$$

$$\min \ G(\mathbf{x}, \mathbf{y}) = \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y}, \tag{1b}$$

where the $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, and $\mathbf{m}$ are the decision variables, and they are all positive integers which must satisfy the following constraints:

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{2a}$$

$$\mathbf{U_1}\mathbf{z} + \mathbf{U_2}\mathbf{m} = \beta, \tag{2b}$$

$$\mathbf{P_1}\mathbf{x} + \mathbf{P_2}\mathbf{y} = \gamma, \tag{2c}$$

$$\mathbf{N_1}\mathbf{z} + \mathbf{N_2}\mathbf{x} + \mathbf{N_3}\mathbf{y} = \zeta. \tag{2d}$$

$$\mathbf{x} \geq \mathbf{0}, \ \mathbf{y} \geq \mathbf{0}, \ \mathbf{z} \geq \mathbf{0}, \ \mathbf{m} \geq \mathbf{0}. \tag{2e}$$

Hereby, $G(\mathbf{x}$ and $\mathbf{y})$ are both linear, but there are no extra requirements on $F(\mathbf{z}, \mathbf{m})$. Meanwhile, $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{b}$, β, γ, and ζ are constant vectors, and $\mathbf{A}$, $\mathbf{U_1}$, $\mathbf{U_2}$, $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{N_1}$, $\mathbf{N_2}$, and $\mathbf{N_3}$ 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 $\mathbf{z}$ and $\mathbf{m}$, while the other only depends on $\mathbf{x}$ and $\mathbf{y}$. 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 during optimization, and we consider subproblem I as the leader, while subproblem II serves as the follower. As mentioned above, we can give the respective mathematical forms as follows:

$$\begin{aligned}
\text{Leader: } \max\ & F(\mathbf{z},\mathbf{m}),\\
\text{s.t., }\ & U_1\mathbf{z}+U_2\mathbf{m}=\beta,\\
& N_1\mathbf{z}+N_2\mathbf{x}+N_3\mathbf{y}=\zeta.\\
& \mathbf{x}\geq\mathbf{0},\ \mathbf{y}\geq\mathbf{0},\ \mathbf{z}\geq\mathbf{0},\ \mathbf{m}\geq\mathbf{0}. \qquad (3)\\
\text{Follower: } \min\ & G(\mathbf{x},\mathbf{y})=\mathbf{c}_1^T\mathbf{x}+\mathbf{c}_2^T\mathbf{y},\\
\text{s.t., }\ & A\mathbf{x}\leq\mathbf{b},\\
& P_1\mathbf{x}+P_2\mathbf{y}=\gamma,\\
& N_1\mathbf{z}+N_2\mathbf{x}+N_3\mathbf{y}=\zeta.\\
& \mathbf{x}\geq\mathbf{0},\ \mathbf{y}\geq\mathbf{0},\ \mathbf{z}\geq\mathbf{0},\ \mathbf{m}\geq\mathbf{0}. \qquad (4)
\end{aligned}$$

Apparently, comparing with the original model, both problems (3) and (4) have relatively small scales. It is relatively easy for solvers to optimize them sequentially. Owing to the definitions of leader and follower, we call such approach as a *game-based decomposition*.

Optimization and convergence

Based on the decomposition and definitions above, we can naturally convert the optimization of the original problem into finding the equilibrium between leader (3) and follower (4). We consider the following alternative and iterative process to optimize leader and follower and find the equilibrium between them:

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:

$$\begin{aligned}
&\max_{\tilde{\mathbf{x}},\tilde{\mathbf{y}}}\left\{\max_{\tilde{\mathbf{x}}}\ \mathbf{b}^T\tilde{\mathbf{x}},\ \max_{\tilde{\mathbf{x}}}\ \sigma\tilde{\mathbf{x}},\ \max_{\tilde{\mathbf{y}}}\ \sigma\tilde{\mathbf{y}}\right\}\\
&\text{s.t., } A^T\tilde{\mathbf{x}}\geq\mathbf{c}_1,\quad P_1^T\tilde{\mathbf{x}}=\mathbf{c}_1,\quad N_2^T\tilde{\mathbf{x}}=\mathbf{c}_1\\
&\quad P_2^T\tilde{\mathbf{y}}=\mathbf{c}_2,\quad N_3^T\tilde{\mathbf{y}}=\mathbf{c}_2\\
&\quad \tilde{\mathbf{y}}\geq\mathbf{0},\quad \tilde{\mathbf{x}}\geq\mathbf{0}, \qquad (5)
\end{aligned}$$

where $\sigma=\left[\gamma+(\zeta-N_1\widehat{\mathbf{z}})^T\right]$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ are the dual variables corresponding with $\mathbf{x}$ and $\mathbf{y}$, respectively, and $\widehat{\mathbf{z}}$ is fixed by leader in the last loop. Assuming that $\tilde{\mathbf{x}}^*$ and $\tilde{\mathbf{y}}^*$ are the optimal solutions of (5), we can add the following constraints to problem (3):

$$F(\mathbf{z},\mathbf{m})+\alpha\geq\max\left\{\mathbf{b}^T\mathbf{x}^*,\sigma\tilde{\mathbf{x}}^*,\sigma\tilde{\mathbf{y}}^*\right\}, \qquad (6)$$

where α is a hyperparameter with corresponding dimension and works as an interface. From optimization perspective, Constraint (6) serves as a heuristic bound and it is apparently additional constraints for leader. It can be understood as that we only *merge* the intersections of all the dual forms. In practice, they can *narrow the search space* and accelerate the convergence with a high probability. Solid lines in Fig. 3 show the results of heuristic constraints. It is observed that in a shorter time, heuristic constraints lead to larger $F(\mathbf{z},\mathbf{m})$ and smaller $G(\mathbf{x},\mathbf{y})$. Thus, we argue that the heuristic constraints can lead to improvements on both the convergence speed and solution quality.

Algorithm

Based on above discussions, we can give the algorithmic flow for proposed game-based decomposition in Algorithm 1.

Algorithm 1 Game-based decomposition algorithm

1: **Input:** raw optimization problem (1), tolerance value ϵ, 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 $F(\mathbf{z}, \mathbf{m})$ and ordinate is $G(\mathbf{x}, \mathbf{y})$. They change along the given directions. Based on the settings, higher quality solutions mean larger $F(\mathbf{z}, \mathbf{m})$ but smaller $G(\mathbf{x}, \mathbf{y})$. 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 $F(\mathbf{z}, \mathbf{m})$ increases and $G(\mathbf{x}, \mathbf{y})$ 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 subproblems, 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.

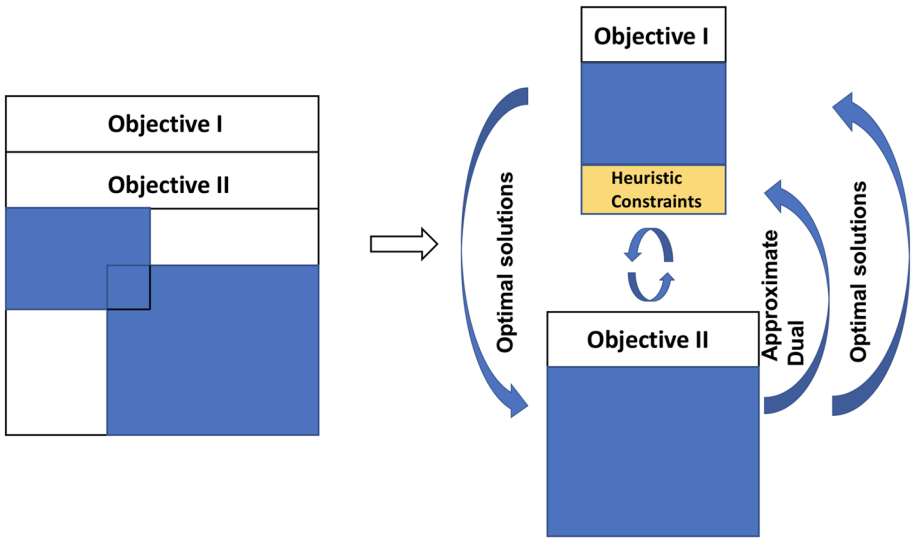

Fig. 2 Graphical illustration for the flow of game-based decomposition

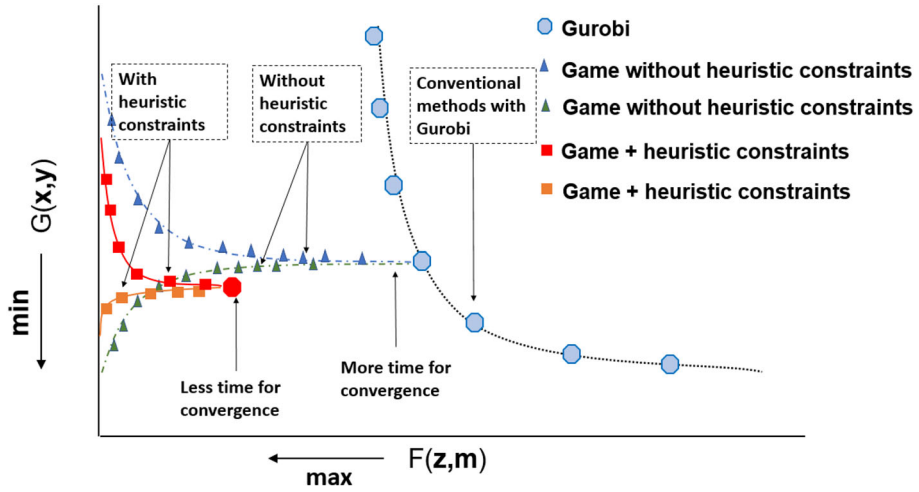

Fig. 3 Results of the game-based algorithm and heuristic constraints. Abscissa is $F(\mathbf{z}, \mathbf{m})$, while ordinate is $G(\mathbf{x}, \mathbf{y})$. The dotted curve denotes the solutions given by Gurobi in a weight-sum way directly. The dashed and solid lines are from our algorithm with and 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

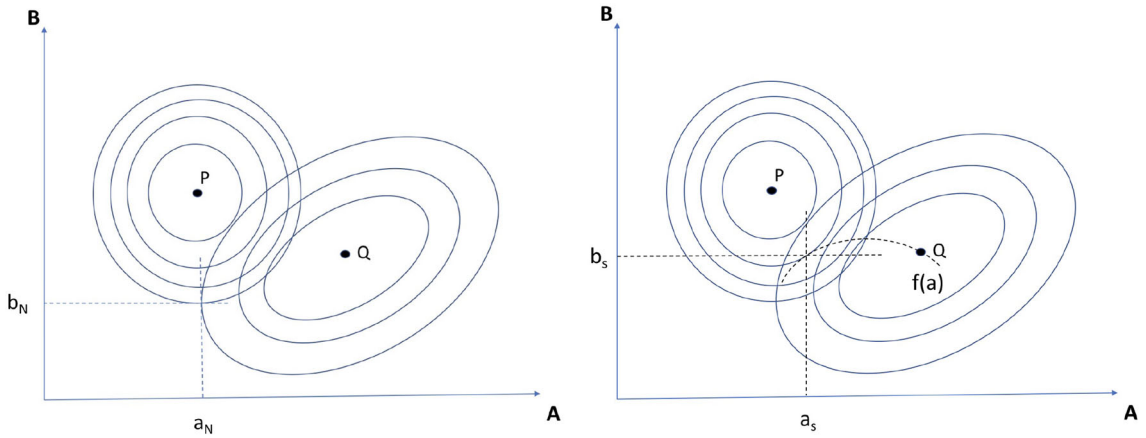

Fig. 4 Graphical illustration for the flow of game-based decomposition

(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:

$$J_1(x, u_1, u_2) = \frac{1}{2}\int_0^\infty r_1(x, u_1, u_2)\mathrm{d}t,$$
$$J_2(x, u_1, u_2) = \frac{1}{2}\int_0^\infty r_2(x, u_1, u_2)\mathrm{d}t,$$

where

$$r_1(x,u_1,u_2) = x^T Q_1 x + u_1^T R_{11} u_1 - u_2^T R_{12} u_2,$$
$$r_2(x,u_1,u_2) = x^T Q_2 x - u_1^T R_{21} u_1 + u_2^T R_{22} u_2,$$

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:

$$J_1^*(x,u_1,u_2) = \min_{u_1} \frac{1}{2}\int_t^\infty r_1(x,u_1,u_2)\mathrm{d}t, \tag{7}$$

and the *follower's Hamiltonian* is:

$$\mathcal{H}_{u_1} = r_1(x,u_1,u_2) + (\nabla J_1)^T (Ax + B_1u_1 + B_2u_2). \tag{8}$$

Thus, the optimal controller can be obtained as $\frac{\partial \mathcal{H}_{u_1}}{\partial u_1} = 0$, that is:

$$u_1^* = -\frac{1}{2}(R_{11})^{-1} B_1^T \nabla J_1. \tag{9}$$

Since the follower's optimal solutions can affect the optimization of leader, the optimal solutions of leader can be expressed as:

$$J_2^*(x,J_1^*,u_2) = \min_{u_2} \frac{1}{2}\int_t^\infty r_2(x,J_1^*,u_2)\mathrm{d}t, \tag{10}$$

with $r_2(x,J_1^*,u_2) = r_2(x,u_1,u_2)\Big|_{u_1=J_1^*}$. Utilizing the gradient of follower's Hamiltonian:

$$\frac{\partial \mathcal{H}_{u_1}}{\partial x} = (Q_1 + Q_1^T)x + A^T \nabla J_1,$$

we can obtain the optimal cost function of leader:

$$J_2^*(x,J_1^*,u_2) = \min_{u_2} \frac{1}{2}\int_t^\infty \left[x^T Q_2 x + \left(u_1^*\right)^T R_{21} u_1^* + u_2^T R_{22} u_2 + \lambda \frac{\partial \mathcal{H}_{u_1}}{\partial x}\right]\mathrm{d}t.$$

Set $r_2(x,J_1^*,u_2) = x^T Q_2 x + \left(u_1^*\right)^T R_{21} u_1^* + u_2^T R_{22} u_2 + \gamma \frac{\partial \mathcal{H}_{u_1}}{\partial x}$, we have the *leader's Hamiltonian* as:

$$\mathcal{H}_{u_2} = \gamma\left[(Q_1 + Q_1^T)x + A^T \nabla J_1\right] + r_2(x,J_1^*,u_2) + (\nabla J_2)^T (Ax + B_1 J_1^* + B_2 u_2), \tag{11}$$

with γ as the real vector.[1] To get the optimal controller, we have $\frac{\partial \mathcal{H}_{u_2}}{\partial u_2} = 0$, which leads to:

$$u_2^* = -\frac{1}{2}(R_{22})^{-1} B_2^T \nabla J_2. \tag{12}$$

Obviously, we also have:

$$\nabla_u \mathcal{H}_{stack} = \left(\frac{\partial \mathcal{H}_{u_1}}{\partial u_1}, \frac{\partial \mathcal{H}_{u_2}}{\partial u_2}\right),$$

and gradients of Hamiltonians for leader and follower also satisfy:

$$\langle \xi(u_1,u_2), (\nabla_u \mathcal{H}_{stack})^T \rangle = 0,$$

with $\xi(u_1,u_2) = (2R_{11}u_1, 2R_{22}u_2)$ and $B_1u_1 + B_2u_2 = 0$, 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:

$$\begin{aligned} &\min \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} + \mathbf{h}^T\mathbf{z}, \\ &\text{s.t.,}\ A\mathbf{x} \le \mathbf{b}, \\ &\quad M\mathbf{x} + N\mathbf{y} + U\mathbf{z} \ge \mathbf{v}, \end{aligned} \tag{13}$$

to show why we can add a constraint to leader with the optimal solutions and dual form of follower. Hereby, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are the decision variables, $\mathbf{c}$ and $\mathbf{d}$ 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:

$$\begin{aligned} &\min \mathbf{c}^T\mathbf{x} + \alpha, \\ &\text{s.t.,}\ A\mathbf{x} \le \mathbf{b}, \\ &\quad \alpha \ge \alpha_{down}, \end{aligned} \tag{14}$$

and the follower:

$$\begin{aligned} &\min \mathbf{d}^T\mathbf{y} + \mathbf{h}^T\mathbf{z}, \\ &\text{s.t.,}\ N\mathbf{y} + U\mathbf{z} \le \mathbf{v} - M\widehat{\mathbf{x}}, \end{aligned} \tag{15}$$

[1] Considering the optimization process, it serves as a Lagrangian multiplier.

where $\widehat{\mathbf{x}}$ 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:

$$\begin{aligned} &\min\ [\mathbf{v} - M\widehat{\mathbf{x}}]^T\ (\tilde{\mathbf{y}} + \tilde{\mathbf{z}})\,, \\ &\text{s.t.},\ N^T\tilde{\mathbf{y}} \le \mathbf{d}, \\ &\quad U^T\tilde{\mathbf{z}} \le \mathbf{h}, \\ &\quad \tilde{\mathbf{y}} \ge 0, \end{aligned} \tag{16}$$

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

$$\begin{aligned} &\min_x \left\{ \mathbf{c}^T\mathbf{x} + \min_y \left\{ \mathbf{d}^T\mathbf{y} + \mathbf{h}^T\mathbf{z} \middle| N^T\tilde{\mathbf{y}} \le \mathbf{d}, U^T\tilde{\mathbf{z}} \le \mathbf{h} \right\} \right\} \\ &= \min_x \left\{ \mathbf{c}^T\mathbf{x} + \max \left\{ \max_{\tilde{y}} \left\{ [\mathbf{v} - M\widehat{\mathbf{x}}]^T\ \tilde{\mathbf{y}} \right\}, \right. \right. \\ &\quad \left. \left. \max_{\tilde{z}} \left\{ [\mathbf{v} - M\widehat{\mathbf{x}}]^T\ \tilde{\mathbf{z}} \right\} \right\} \right\} \\ &\le \min_x \left\{ \mathbf{c}^T\mathbf{x} + \max \left\{ [\mathbf{v} - M\widehat{\mathbf{x}}]^T\ \tilde{\mathbf{y}}^*, [\mathbf{v} - M\widehat{\mathbf{x}}]^T\ \tilde{\mathbf{z}}^*, \right\} \right\} \end{aligned}$$

where $\tilde{\mathbf{y}}^*$ and $\tilde{\mathbf{z}}^*$ 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, $i \in I = I_P \cup I_{AI}$ 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_{AI} represent the sets of semi-finished and finished products, respectively. Other related notations are presented in Table 1.

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:

$$\max\ F(z,m) = \sum_{i\in I, t\in T} \frac{\sum_{p\in P_i^{\downarrow}} \left(z_{i,p,t} - m_{i,t-1}\right)}{D_{i,t} + F_{i,t}} \tag{17}$$

$$\min\ G(x,s,v) = g_1(x) + g_2(s) + g_3(x,s,v), \tag{18}$$

Table 1 Variables and notations

Variables	Notations
$z_{i,p,t}$	Supply quantity of item i of plant p
$s_{i,p,t}^{p'}$	Transportation quantity of item i of plant p at time t
$x_{i,p,t}$	Planning production quantity of item i of plant p at time t
$m_{i,t}$	Delay quantity of item i at time t
$r_{i,p,t}^{i'}$	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
$in_{i,p,t}$	inbound constraint of item i of plant p at time t
$\text{out}_{i,p,t}$	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
$P_i^{\downarrow}$	Plants set of item i
$P_{i,p}^{\downarrow}$	Plants set of item i of plant p
$LT_p^{p'}$	Lead time
$\text{PC}_{i,p}$	Processing cost of item i of plant p
$\text{PT}_{i,p}$	Processing time of item i of plant p
$\text{TC}_{i,p}^{p'}$	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
$B_{i,p,t}^{i'}$	Cost for unit quantity
$U_{i,p,c}$	Unit occupancy of item i of plant p
$\text{CAP}_{p,t,s,c}$	Upper limit for production
$\text{MLS}_{i,p,t}$	Minimal production quantity of item i of plant p at time t
$\text{PAIR}_{i,p}^{p'}$	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
Δ_1, Δ_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:

$$g_1(x) = \sum_{i\in I_{AI}} \sum_{p\in P^{\downarrow}} \text{PC}_{i,p} \cdot \left(\sum_t \text{PM}_i \cdot x_{i,p,t} \right)$$

$$g_2(s) = \sum_{i\in I} \sum_{p\in P^{\downarrow}} \left[\sum_{p\in\{P^{\downarrow}\setminus p'\}} \text{TC}_{i,p}^{p'} \cdot \left(\sum_t s_{i,p,t}^{p'} \right) \right]$$

$$\begin{aligned} g_3(x,s,v) = &\sum_{i\in I} \sum_{p\in P^{\downarrow}} \text{HC}_{i,p} \cdot \left(\sum_t v_{i,p,t} \right) \\ &+ \sum_{i\in I} \sum_{p\in P^{\downarrow}} \text{HC}_{i,p} \cdot \left(LT_{p'}^{p} \cdot \sum_t s_{i,p',t}^{p} \right) \\ &+ \sum_{i\in I_{AI}} \sum_{p\in P^{\downarrow}} \text{HC}_{i,p} \cdot \left(\text{PT}_{i,p} \cdot \sum_{t\in T} \text{PM}_i \cdot x_{i,p,t} \right). \end{aligned}$$

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

$$\sum_{i\in I_s^{\uparrow}} U_{i,p,c}\cdot\left(\mathrm{PM}_i\cdot x_{i,p,t}\right)\le CAP_{p,t,s,c} \tag{19}$$

$$\mathrm{PM}_{i'}\cdot x_{i',p,t}=PAIR_{i,p}^{i'}\cdot\left(\mathrm{PM}_i\cdot x_{i,p,t}\right),\quad i'\neq i \tag{20}$$

$$\mathrm{PM}_i\cdot x_{i,p,t}\in\{0,\ MLS_{i,p,t},\ MLS_{i,p,t}+1,\ldots\} \tag{21}$$

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):

$$m_{i,t}=\begin{cases} M_i, & t=0\\ m_{i,t-1}+D_{i,t}-\sum_{p\in P_i^{\downarrow}} z_{i,p,t}, & \text{otherwise.}\end{cases} \tag{22}$$

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

$$v_{i,p,t}=\begin{cases} V_{i,p}, & t=0\\ v_{i,p,t-1}+in_{i,p,t}-\mathrm{out}_{i,p,t} & \text{otherwise.}\end{cases} \tag{23}$$

$$\begin{aligned} in_{i,p,t}=&\sum_{p'\in P_{i,p}^{\downarrow}} s^{p'}_{i,p,t-LT_p^{p'}-\Delta_1}\\ &+\mathrm{PM}_i\cdot x_{i,p,t-\mathrm{PT}_{i,p}-\Delta_2}+PO_{i,p,t}. \end{aligned} \tag{24}$$

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

$$\begin{aligned} \mathrm{out}_{i,p,t}=&\sum_{p'\in P_{i,p}^{\uparrow}} s^{p}_{i,p',t}+\Bigg[\sum_{i'\in\{I_{i,p}^{\uparrow}\cap I_p^{\uparrow}\}}\left(B_{i,p,t}^{i'}\cdot\mathrm{PM}_i\cdot x_{i',p,t}\right)\\ &+z_{i,p,t}-\sum_{i'\in I_i^{R\uparrow}} r_{i,p,t}^{i'}\Bigg]+\sum_{i'\in I_i^{R\downarrow}} r_{i',p,t}^{i} \end{aligned} \tag{25}$$

$$\sum_{i'\in\{I_{i,p}^{\uparrow}\cap I_p^{\uparrow}\}}\widehat{x_{i'}^{r}}+z_{i,p,t}-\sum_{i'\in I_i^{R\uparrow}} r_{i,p,t}^{i'}\le 0, \tag{26}$$

with $\widehat{x_{i'}^{r}}=B_{i,p,t}^{i'}\cdot\left(\mathrm{PM}_i\cdot x_{i',p,t}\right)$. 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 ϵ, iteration counter $\gamma=1$, and hyperparameter α_{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:

$$\begin{aligned} &\max F(z,m)+\alpha,\\ &\text{s.t. Constraint delay, inventory, outbound and inbound}\\ &\qquad \alpha\ge\alpha_{\text{initial}},\qquad 0\le x\le x^{up}, \end{aligned} \tag{27}$$

  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 $\mathrm{out}=\mathrm{out}^{(\gamma)}$.
- **Step** 3**: Convergence check**.
  - **Step** 3.1. Compute an upper bound (UB) as below:

$$\begin{aligned} \Omega_{upper}^{(\gamma)}=&\beta_1^{(\gamma)}\left[g_1(x^{(\gamma)})+g_2(s^{(\gamma)})+g_3(x^{(\gamma)},s^{(\gamma)},v^{(\gamma)})\right]\\ &+\beta_2^{(\gamma)}\frac{\sum_{p\in P_i^{\downarrow}}\left(z_{i,p,t}^{(\gamma)}-m_{i,t-1}^{\gamma}\right)}{D_{i,t}+F_{i,t}}, \end{aligned}$$

    where $\beta_1^{(\gamma)}$ and $\beta_2^{(\gamma)}$ are two hyperparameters.
  - **Step** 3.2. Compute a lower bound (LB) as below:

$$\Omega_{lower}^{(\gamma)}=\beta_1^{(\gamma)}\alpha^{(\gamma)}+\beta_2^{(\gamma)}\frac{\sum_{p\in P_i^{\downarrow}}\left(z_{i,p,t}^{(\gamma)}-m_{i,t-1}^{\gamma}\right)}{D_{i,t}+F_{i,t}}.$$

  - **Step** 3.3. If $\Omega_{upper}^{(\gamma)}-\Omega_{lower}^{(\gamma)}<\epsilon$, 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 ($\mathcal{I}$) of follower $G(x,s,v)$, according to (5).
- **Step** 5**: Update Leader**, i.e., resolve Leader (27) with heuristic constraints,

$$\begin{aligned} &\max F(z,m)=\sum_{i\in I,t\in T}\frac{\sum_{p\in P_i^{\downarrow}}\left(z_{i,p,t}-m_{i,t-1}\right)}{D_{i,t}+F_{i,t}}+\alpha,\\ &\text{s.t. } m_{i,t}=\begin{cases} M_i, & t=0\\ m_{i,t-1}+D_{i,t}-\sum_{p\in P_i^{\downarrow}} z_{i,p,t}, & \text{otherwise}\end{cases}\\ &\qquad \alpha\ge\alpha_{\text{initial}},\quad 0\le x\le x^{up},\\ &\qquad g_1(x^{(k)})+g_2(s^{(k)})+g_3(x^{(k)},s^{(k)},v^{(k)}) \end{aligned}$$

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

$$+\sum_{i\in\mathcal{I}}\lambda_{1,i}^{(k)}\left(z_i-z_i^{(k)}\right)+\sum_{i\in\mathcal{I}}\lambda_{2,i}^{(k)}\left(m_i-m_i^{(k)}\right)\leq\alpha,$$

where $\lambda_{1,i}^{(k)}$ and $\lambda_{1,i}^{(k)}$ $(k=1,\cdots,\gamma)$ 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]

[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.

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 days, 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 billion 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-

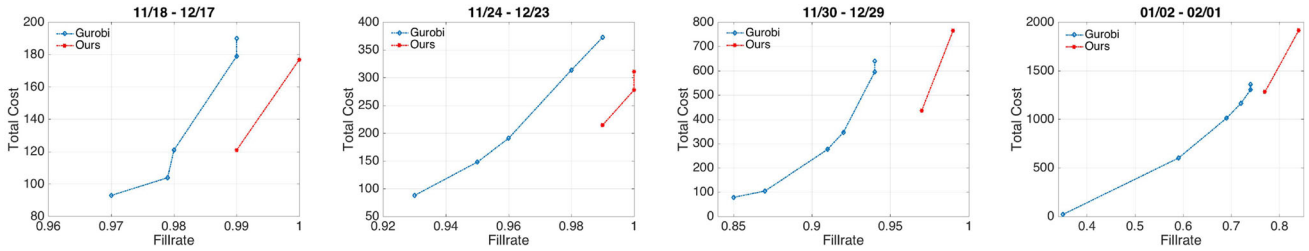

Fig. 5 Evaluation on Dataset I-1

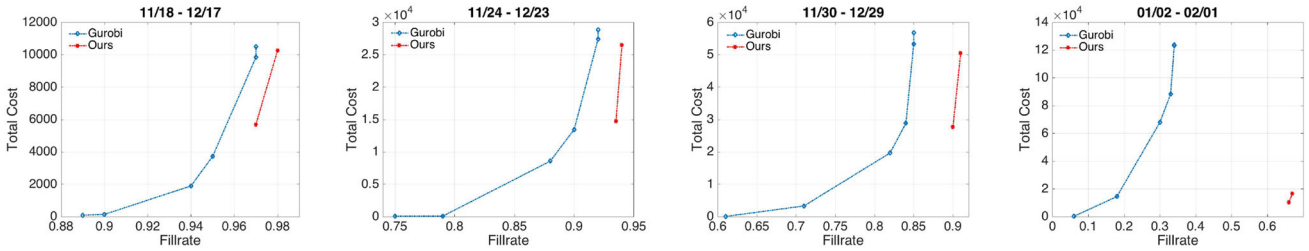

Fig. 6 Evaluation on Dataset I-2

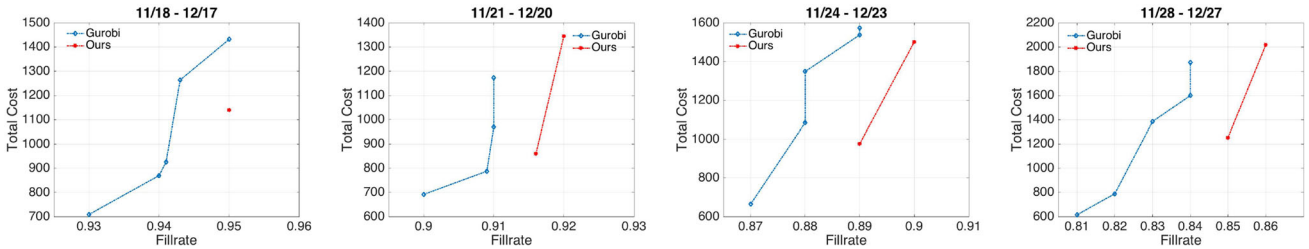

Fig. 7 Evaluation on Dataset II-1

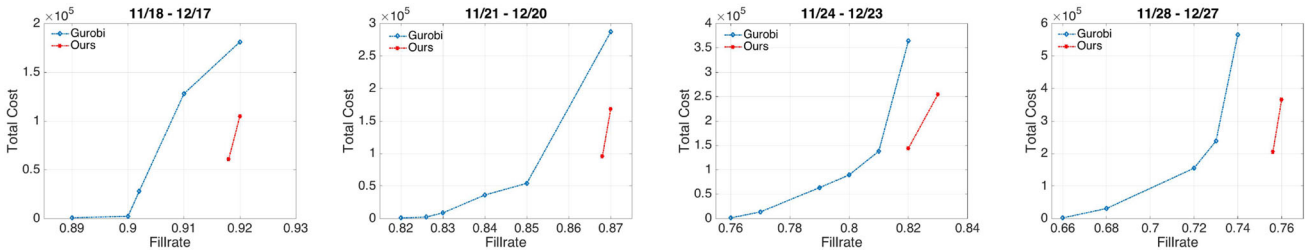

Fig. 8 Evaluation on Dataset II-2

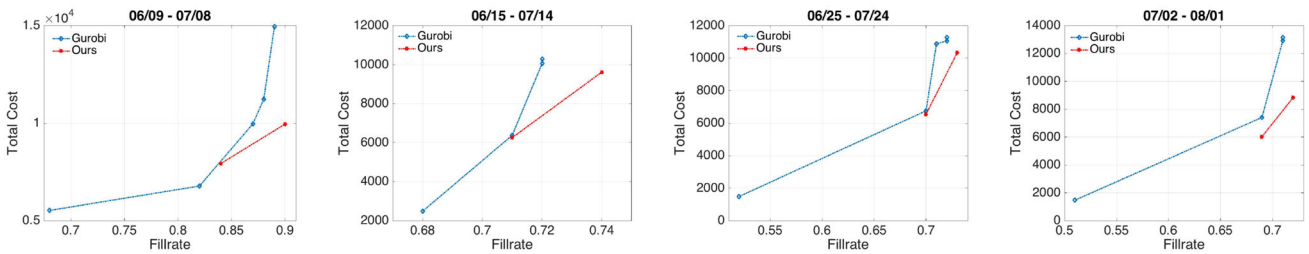

Fig. 9 Evaluation on Dataset III

rithm's efficiency decreases much slower than Gurobi's as problem scale increases.

Different from other decomposition algorithms, our algorithm can deal with non-strict blocked problems. Our major

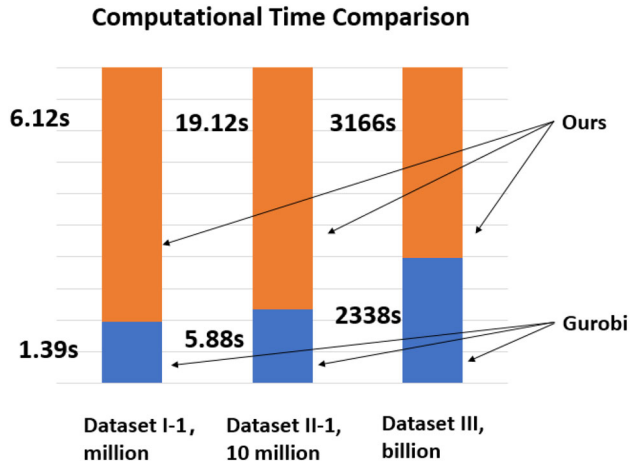

Fig. 10 Computational time comparison between our algorithm and Gurobi on Datasets I-1 (11/24-12-23), II-1 (11/24-12/23), and III (06/09-07/08)

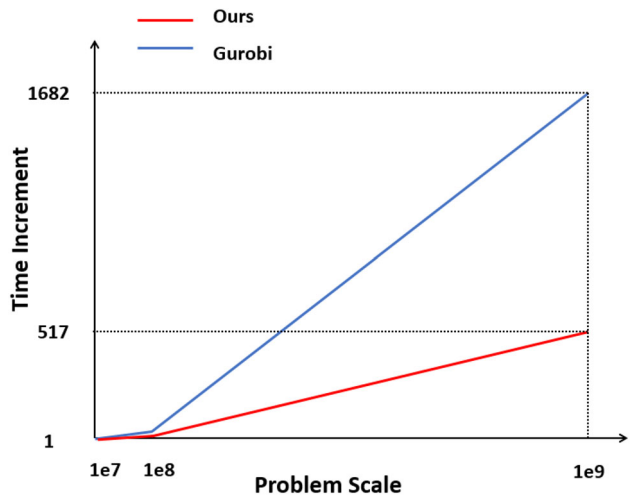

Fig. 11 Comparison on time increments as problem scale increases. It is shown that our algorithm's efficiency decreases much slower than Gurobi's when the problem scale increases

contributions include: (1) a new decomposition algorithm inspired by game, which is different from previous works and can deal with non-strict blocked problems; (2) new optimization process, which overcomes the large scale and converge to a solution; (3) 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 algorithm 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).

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, Minguez R, Garcia-Bertrand R (2006) Decomposition techniques in mathematical programming: engineering and science applications. Springer, Berlin
6. Fischetti M, Ljubić I, Sinnl M (2016) Redesigning benders 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