Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Distributed Job Shop Scheduling using Consensus Alternating Direction Method of Multipliers

Toshiyuki Miyamoto [*] **Toyohiro Umeda** [**] **Shigemasa Takai** [*]

[*] *Osaka University, Suita, 565-0871 Japan (e-mail: miyamoto@eei.eng.osaka-u.ac.jp).*
[**] *Kobe Steel, Ltd., Kobe, 651-2271 Japan*

Abstract: Scheduling problems belong to NP-hard and are not easily solved in large systems. In recent years, the development of optimization methods in multi-agent systems has been remarkable. In this paper, we consider a large-scale system as a multi-agent system and discuss a method of solving a scheduling problem using consensus among agents. We propose a distributed method using the alternating direction method of multipliers and evaluate the method using a small-scale instance of the scheduling problem.

Keywords: distributed scheduling, consensus, optimization, ADMM, job shop scheduling

1. INTRODUCTION

Since the manufacturing cost in large-scale production systems such as steel manufacturing processes is enormous, optimization on its operation plan is important from the viewpoint of cost reduction and effective use of resources. However, there is a practical problem in terms of calculation time on obtaining the optimal schedule or optimizing the entire manufacturing process. Even if an optimal schedule for the entire manufacturing process was obtained, keeping the operation on the schedule is extremely difficult due to the fact that there exists some gap between the model and the actual plant and there exist various variable factors that cannot be avoided in actual operation. For this reason, even if a manufacturing plan can be obtained as a guideline, the operation is often left to the field. As a result, some concerns such as unnecessary job accumulation, the material shortage at the neck equipment, equipment stoppage due to overflow of the storage location (buffer), delay in delivery, and a shortfall in production happen. On the other hand, the environment where equipment and logistics information in the factory can be acquired in real-time is being improved by introducing IoT. In addition, an environment has been rapidly introduced in which operation and transport instructions can be transmitted directly to the equipment in real-time. Therefore, in this paper, we consider a multi-agent approach that guides the entire system in the desired direction while satisfying local constraints by exchanging information between agents.

As an important problem in multi-agent systems, the consensus problem exists. The consensus problem is an optimization problem aiming at minimizing the sum of the objective functions of agents while keeping some variables common to multiple agents. Various applications of the consensus problem (Ren et al. (2005)) have been reported such as scheduling problems (Moore and Lucarelli (2007)), QoS fairness control of computing resources (Hayashi et al. (2010)), cooperative tracking by camera sensor network (Segawa et al. (2015)), source identification (Hayashi and Takai (2017)), economic dispatch problem in smart grids (Nguyen et al. (2018)).

In this paper, we propose to apply the distributed algorithm of the consensus problem to the job shop scheduling problem as the first step toward the construction of an autonomous distributed scheduling method for large-scale production systems. After describing the consensus problem and its distributed algorithm in Sect.2, we propose an application method for the job shop scheduling problem in Sect.3. Section 4 explains the proposed method using a simple example and shows the results of computer experiments.

2. CONSENSUS PROBLEM AND ITS DISTRIBUTED ALGORITHM

Let X be a convex set, I be a set of agents, $f_i : X \to \mathbb{R}$ be a closed, proper, and convex objective function of agent $i \in I$. Then, the consensus problem can be formulated as follows:

$$\text{(CP1)} \quad \min_{x \in X} \sum_{i \in I} f_i(x) \tag{1}$$

In the consensus problem, the sum of objective functions is minimized while consensus on shared common variable x is achieved.

Distributed optimization is achieved by repeating optimization by each agent and information exchange between agents. If there are restrictions on the communication among agents, the restriction structure is represented by a graph $G = (I, E)$, where $E \subseteq I^2$. In this paper, we assume that the graph is connected and undirected (meaning that a communicable partner can communicate in both directions). Let $N_i = \{i' \in I \mid (i, i') \in E\}$ be the set of neighboring agents of agent i, the domain of variables for agent $i \in I$ be X_i, and $X = \cap_{i \in I} X_i$. Then, the consensus problem can be rewritten as follows:

$$\text{(CP2)} \quad \min_{x_i \in X_i} \quad \sum_{i \in I} f_i(x_i) \tag{2}$$

$$\text{s.t.} \quad x_i = x_{i'}, \quad i' \in N_i, i \in I \tag{3}$$

The Alternating Direction Method of Multipliers (ADMM) (Boyd et al. (2010)) is a solution to the convex optimization problem, by alternatively updating variables by optimizing the augmented Lagrangian function of the objective function. If a problem can be decomposed into subproblems of each agent's variable set, it can be used as a distributed optimization technique. The ADMM algorithm for the consensus problem on the graph is as follows (Mateos et al. (2010); Chang et al. (2015)), where a_i is a parameter of agent i, c is a parameter common to agents, and the number of right shoulders indicates the number of iterations:

(1) Initialize $a_i^{(0)}$ and $x_i^{(0)}$ for each agent i and let $k := 0$.
(2) Exchange x_i with neighboring agents and let $k := k + 1$.
(3) Update a_i and x_i for each agent i as follows:

$$a_i^{(k)} = a_i^{(k-1)} + c \sum_{i' \in N_i} (x_i^{(k-1)} - x_{i'}^{(k-1)}) \tag{4}$$

$$x_i^{(k)} = \arg \min_{x_i \in X_i} \left\{ f_i(x_i) + x_i^{\mathrm{T}} a_i^{(k)} \right.$$
$$\left. + c \sum_{i' \in N_i} \left\| x_i - \frac{x_i^{(k-1)} + x_{i'}^{(k-1)}}{2} \right\|_2^2 \right\} \tag{5}$$

(4) Repeat steps (2) and (3) while a stopping condition is not satisfied.

At step (1) in the algorithm, there is a condition on $a_i^{(0)}$. An easy way to satisfy the condition is setting 0 to each $a_i^{(0)}$ (Chang et al. (2015)).

At step (2), agents exchange decision variable values.

The step (3) can be executed in parallel for each agent. Each agent updates the dual variable a_i by eq. (4), then, he or she updates the decision variable x_i by solving optimization problem in eq. (5).

The algorithm converges to an optimal solution with $k \to \infty$. Actually, it is repeated until a stopping condition is satisfied as in step (4). As the stopping criteria, the primal residual $r^{(k)}$ and dual residual $s^{(k)}$ given by the following equation are used.

$$r^{(k)} = \sum_{i \in I} \sum_{i' \in N_i} \| x_i^{(k)} - x_{i'}^{(k)} \|_2^2 \tag{6}$$

$$s^{(k)} = \sum_{i \in I} \| x_i^{(k)} - x_i^{(k-1)} \|_2^2 \tag{7}$$

3. APPLICATION TO JOB-SHOP SCHEDULING

In this paper, we propose to apply the ADMM algorithm to Job-shop Scheduling Problem (JSP). Note that since JSP is a non-convex optimization problem, optimality is not guaranteed. Furthermore, convergence by the proposed method is not guaranteed.

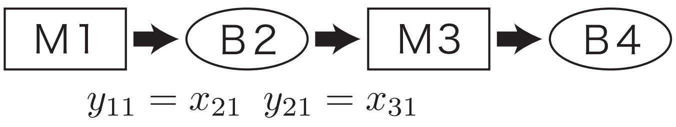

Fig. 1. Example system

A production system consists of machines and buffers, where transport equipment is considered buffers. The machine and the buffer are collectively called equipment. We consider a multi-agent system where an agent exists for each piece of equipment and each agent independently determines the schedule of the piece of equipment. At this time, if there is a discrepancy between the end time and the start time of consecutive pieces of equipment, execution of the schedule is not possible. Therefore, we formulate JSP as the consensus problem on these times.

Let I be the set of equipment, J be the set of jobs. Each job $j \in J$ visits every piece of equipment in a beforehand determined order and the $\imath$-th equipment of job j is denoted by $m_{j,\imath}$.

The decision variables are the process start time and process end time. The processing start time of job $j \in J$ in equipment $i \in I$ is x_{ij}, and the processing end time is y_{ij}. When the equipment i is a buffer, x_{ij} represents the time when the job j is placed in the buffer, and y_{ij} represents the time it is taken out of the buffer. The decision variables vector of equipment i is $x_i = [x_{i,1}, \ldots, x_{i,|J|}, y_{i,1}, \ldots, y_{i,|J|}]^{\mathrm{T}}$ and its domain is X_i. It is assumed that the processing time constraint on the machine, the number of jobs that can be processed simultaneously, the job ready time constraint, the delivery time constraint, the transfer time constraint between machines, the buffer capacity constraint, etc. are all expressed as X_i.

JSP is formulated as a consensus problem as follows because the end time and the start time need to be agreed among consecutive processes.

$$\text{(JSP)} \quad \min_{x_i \in X_i} \quad \sum_{i \in I} f_i(x_i) \tag{8}$$

$$\text{s.t.} \quad y_{m_{j,\imath} j} = x_{m_{j,\imath+1} j},$$
$$\imath \in \{1, \ldots, |I| - 1\}, j \in J \tag{9}$$

4. COMPUTATIONAL EXPERIMENTS

4.1 Example

Consider a production system consisting of two machines (M1, M3), an intermediate buffer (B2), and a finished product buffer (B4) shown in Fig. 1. All jobs are placed in buffer B2 after processing on machine M1. Jobs sent from buffer B2 to machine M3 are placed in buffer B4 after processing on machine M3. The processing times on machines M1 and M3 are given constant and buffer B2 is given a lower limit for the time to be placed. The job has a due date, and a penalty occurs if the time placed in buffer B4 exceeds the due date. Table 1 shows the ready time, process time, and due date of jobs.

In the proposed method, agents exist in M1, B2, M3, and B4. Each agent makes a schedule while agreeing a time with neighboring agents. For example, y_{11} is the time in

Table 1. Ready time, process time, and due-date of jobs

Job	ready time	process time			due-date
		M1	B2	M3	B4
		p_{1j}	p_{2j}	p_{3j}	d_j
j_1	0	3	5	4	18
j_2	0	6	1	2	10
j_3	0	2	2	8	16

Table 2. Optimal solution

(a) x^*_{ij}

i	j 1	2	3
1	2	5	0
2	5	11	2
3	14	12	4
4	18	14	12

(b) y^*_{ij}

i	j 1	2	3
1	5	11	2
2	14	12	4
3	18	14	12

the schedule of machine M1 when machine M1 places job 1 in the buffer, x_{21} is the time in the schedule of buffer B2 when job 1 is placed in buffer B2 , y_{21} is the time in the schedule of buffer B2 when job 1 is taken out from buffer B2, and x_{31} is the time in the schedule of machine M3 when machine M3 takes out job 1 from the buffer. The agents need to create a schedule while agreeing that $y_{11} = x_{21}$ between machine M1 and buffer B2 and $y_{21} = x_{31}$ between buffer B2 and machine M3.

Let $I = \{1, 2, 3, 4\}$ be the set of equipment, $J = \{1, 2, 3\}$ be the set of jobs, p_{ij} be the processing time of job j in equipment i, and d_j be the due date of job j, where the processing time in the buffer represents the time interval required before moving from the previous machine to the next machine.

The above scheduling problem is formulated as a centralized optimization problem as follows:

$$\min \sum_{j\in J} \max\{0, x_{4j} - d_j\} \tag{10}$$

$$\text{s.t.} \quad x_{1j} \ge 0, j \in J \tag{11}$$

$$y_{ij} - x_{ij} = p_{ij}, i \in \{1, 3\}, j \in J \tag{12}$$

$$y_{ij} - x_{ij} \ge p_{ij}, i \in \{2\}, j \in J \tag{13}$$

$$y_{ij_1} \le x_{ij_2} \vee y_{ij_2} \le x_{ij_1} i \in \{1, 3\}, j_1 \ne j_2 \tag{14}$$

$$y_{ij} = x_{i+1,j}, i \in \{1, 2, 3\}, j \in J \tag{15}$$

The objective function was set to minimize the sum of delays to the due date (tardiness). (11) is a ready time constraint, and (12) and (13) are processing time constraints. The processing time constraint in the buffer is an inequality constraint because it represents the minimum interval. (14) is a constraint on the number of jobs that can be processed simultaneously on a machine, and (15) is a consensus constraint.

The objective function value of the optimal solution of the example shown in Table 1 is 4, and values of the optimal solution x^*_{ij} and y^*_{ij} are shown in Table 2.

The objective function (10) can be seen as $\sum_{i\in I} f_i(x_i)$ when $f_1(x_1) = f_2(x_2) = f_3(x_3) = 0$, and $f_4(x_4) = \sum_{j\in J} \max\{0, x_{4j} - d_j\}$. The constraints (11) – (14) are local, i.e., they define X_i. The constraint (15) is the consensus constraint. Thus, the scheduling problem has the form of CP2.

4.2 Formulation for distributed optimization

Since the scheduling problem has the form of CP2, the ADMM algorithm can be applied. The optimization problem solved by each agent at eq. (5) when the above optimization problem is distributively optimized using ADMM is as follows, where $x_i = [x_{i,1}, \ldots, x_{i,|J|}]^{\mathrm{T}}$, $y_i = [y_{i,1}, \ldots, y_{i,|J|}]^{\mathrm{T}}$.

Machine M1:

$$y_1^{(k)} = \arg\min_{y_1} \left\{ y_1^{\mathrm{T}} a_1^{(k)} + c \left\| y_1 - \frac{y_1^{(k-1)} + x_2^{(k-1)}}{2} \right\|_2^2 \right\}$$

$$\begin{aligned} \text{s.t.} \quad & x_1 \ge 0 \\ & y_1 - x_1 = p_1 \\ & y_{1j_1} \le x_{1j_2} \vee y_{1j_2} \le x_{1j_1}, j_1 \ne j_2 \end{aligned}$$

Buffer B2:

$$\begin{aligned} \begin{bmatrix} x_2^{(k)} \\ y_2^{(k)} \end{bmatrix} = \arg\min_{x_2, y_2} \Bigg\{ & [x_2^{\mathrm{T}}, y_2^{\mathrm{T}}] a_2^{(k)} \\ & + c \left(\left\| x_2 - \frac{x_2^{(k-1)} + y_1^{(k-1)}}{2} \right\|_2^2 \right. \\ & \left. + \left\| y_2 - \frac{y_2^{(k-1)} + x_3^{(k-1)}}{2} \right\|_2^2 \right) \Bigg\} \\ \text{s.t.} \quad & y_2 - x_2 \ge p_2 \end{aligned}$$

Machine M3:

$$\begin{aligned} \begin{bmatrix} x_3^{(k)} \\ y_3^{(k)} \end{bmatrix} = \arg\min_{x_3, y_3} \Bigg\{ & \begin{bmatrix} x_3^{\mathrm{T}} \\ y_3^{\mathrm{T}} \end{bmatrix} a_3^{(k)} \\ & + c \left(\left\| x_3 - \frac{x_3^{(k-1)} + y_2^{(k-1)}}{2} \right\|_2^2 \right. \\ & \left. + \left\| y_3 - \frac{y_3^{(k-1)} + x_4^{(k-1)}}{2} \right\|_2^2 \right) \Bigg\} \\ \text{s.t.} \quad & y_3 - x_3 = p_3 \\ & y_{3j_1} \le x_{3j_2} \vee y_{3j_2} \le x_{3j_1}, j_1 \ne j_2 \end{aligned}$$

Buffer B4:

$$\begin{aligned} x_4^{(k)} = \arg\min_{x_4} \Bigg\{ & \sum_{j\in J} \max\{0, x_{4j} - d_j\} + x_4^{\mathrm{T}} a_4^{(k)} + \\ & c \left(\left\| x_4 - \frac{x_4^{(k-1)} + y_3^{(k-1)}}{2} \right\|_2^2 \right) \Bigg\} \end{aligned}$$

4.3 Experiments

Figs. 2 and 3 show the execution results when parameter $c = 0.5$ and 0.1, respectively, and the decision variables are integers. Fig. 4 shows the result when $c = 0.1$ and the decision variables are real numbers. In all the results, the horizontal axis is the number of iterations and the vertical axis is the variable x_{ij}. The dotted lines in the figures show the optimal solution in Table 2.

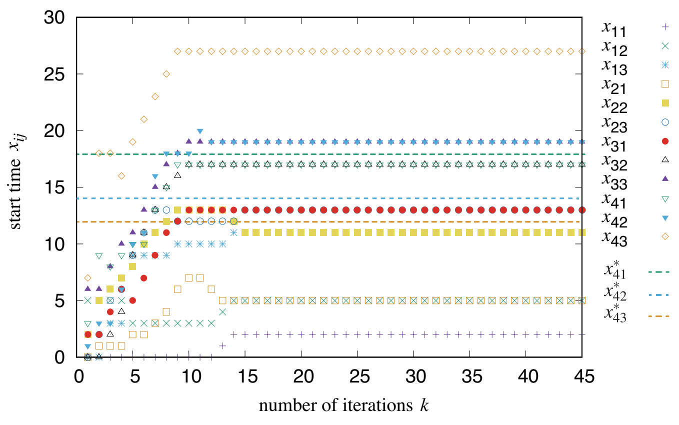

Fig. 2. Change of start time x_{ij} when $c = 0.5$ and decision variables are integers. The algorithm converges, but objective function value is 20.

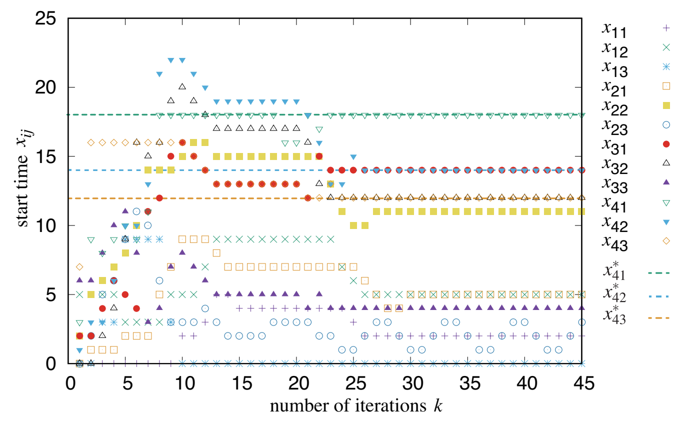

Fig. 3. Change of start time x_{ij} when $c = 0.1$ and decision variables are integers. Objective function value is 4, but the algorithm has not converged.

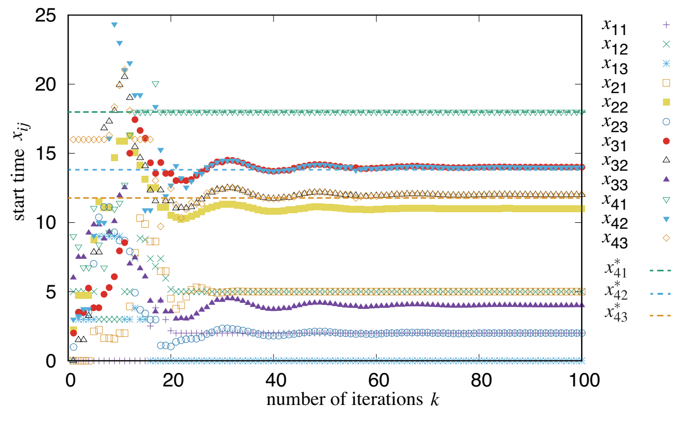

Fig. 4. Change of start time x_{ij} when $c = 0.1$ and decision variables are real numbers. The algorithm converges and objective function value is 4.

In Fig. 2, the algorithm converges, but the objective function value is 20. In Fig. 3, the objective function value is 4, but it has not converged. In Fig. 4, the algorithm converged to the optimal solution.

In our formulation in equation (5), the parameter c is multiplied by the norm that represents the difference form the mean value of the consensus variable in the previous iteration. When the parameter c is large, the ADMM algorithm hurries to converge. As a result, the algorithm goes to bad solution as shown in Fig. 2. When the parameter c is small, it is expected that the ADMM algorithm goes to good solution; however, we need more iterations.

In Fig. 3, the value of x_{23} vibrates. What happens there is as follows. At $k = 29, 32, 35, 38, 41,$ and 44, the consensus was achieved, i.e. the primal residual was zero, however, the dual residual was not enough small. Thus, the stopping condition was not satisfied. Furthermore, the ADMM algorithm can be seen a dynamical system, and thus, it has a kind of inertia. So the algorithm passed by the solution and oscillated near the solution. Because the inertia comes from the second term $(x_i^{\mathrm{T}} a_i^{(k)})$ in equation (5), one possible countermeasures to make stop the algorithm is increasing the parameter c. In fact, varying penalty parameter is an issue in convex optimization (Boyd et al. (2010)).

The vibration of values is not only the case as x_{23} in Fig. 3. Suppose that $x - y = 1$ and we want to achieve a consensus between x and y. The ADMM algorithm updates their value as $x := x - 1$ and $y := y + 1$ to achieve a consensus. Then x and y will exchange their values forever because they are integers.

Another countermeasure against these vibrations is changing the type of variables from integers to real numbers. As shown in Fig. 4, the algorithm converged to the optimal solution. However, due to the slow update of variables, the algorithm requires a large number of iterations when the stopping condition is strict. It is known that increasing the parameter c is effective also in such a case.

In the above three cases, although the variables, which represent start and end times, vibrate, the order of processing jobs is fixed. So another countermeasure is that we watch the order too and stop the iteration when the order is fixed. After fixing the order, we can set the start and end times by forward/backward simulation of the entire system using the fixed order.

If we run the ADMM algorithm in another instance, we can see a case where the order of processing jobs vibrates. One of the reasons that this kind of vibration occurs is that each agent does not have its own objective function in the formulation in Sect.4.2. In the formulation, only buffer B4 has its own objective to minimize the sum of tardiness; other agents have only penalty terms come from the Lagrangian relaxation and do not have their own objective. So even if a vibration on the order occurs in the preceding three agents, these agents have no intention stabilizing the vibration. When we apply the proposed method to a more complicated production system, each agent may have its own objective. We are expecting that the agents' own objective has some effects on stabilizing this kind of vibrations.

5. RELATED WORKS

In this paper, we studied a multi-agent-based scheduling problem. The concept of multi-agent-based scheduling itself is not new. The Intelligent Manufacturing System

(IMS) program[1] started in 1995 as a trilateral cooperative research project among the US, EU, and Japan. We can find agent-based manufacturing in some of early projects such as Next Generation Manufacturing Systems project (Okabe et al. (1998)), (Bunce et al. (1997)) and Holonic Manufacturing Systems project (Van Brussel et al. (1997)).

The scheme of multi-agent-based scheduling can be classified into three types: heuristic-based, meta-heuristic-based, and optimization-based.

In heuristic-based methods, agents exist for each machine and/or job; agents make a decision using classical dispatching rules. Miyamoto et al. (2002) applied context-dependent agents (CDAs) to JSP, where an agent exists for each machine, each agent has a set of dispatching rules and selects one or a couple of rules depending on the context of the production system to decide the next operation. Lee et al. (2012) considered a distributed scheduling where each agent holding a job selects a machine to process its own job and gave bounds for the price of anarchy for some objectives: the makespan, the total congestion time, the total completion time, the maximum tardiness, the total tardiness, and the number of tardy jobs.

In meta-heuristic-based methods, a multi-agent system solves JSP using meta-heuristics. Ennigrou and Ghédira (2008) proposed multi-agent approaches based on the tabu search. Xiong and Fu (2015) proposed a multi-agent scheduling approach using immune system.

In optimization-based methods, distributed optimization methods are used to solve JSP. Gou et al. (1994), Liu et al. (2007), and Xu et al. (2012) use Lagrangian decomposition (LD) (Guignard and Kim (1987)) to solve JSP in a distributed way. When we solve an optimization problem in a distributed way, we have to decompose it into a set of the optimization problem for each agent. LD is often used for such a purpose in many applications, for example, vehicle scheduling (Nishi and Tanaka (2012)) and assignment problem (Hanada and Hirayama (2011)). In LD, global constraints that lie among multiple agents are relaxed and added to the objective function with penalty parameters called Lagrangian multipliers, which is a_i in our formulation (4) and (5). Lagrangian multipliers represent degrees of competitiveness of shared resources. In (Gou et al. (1994)) and (Liu et al. (2007)), job agents try to acquire machine agents to complete the job and LD is used to resolve conflicts among job agents. In (Xu et al. (2012)), LD is used to achieve consensus on start and end times among jobs, so, it is very close to our approach.

The difference between LD and ADMM is how to relax the problem. In LD, Lagrangian is used, whereas augmented Lagrangian is used in ADMM. Therefore if we choose 0 as the value of parameter c, the method described in this paper becomes LD. Comparing to LD, ADMM has better convergence. In fact, in convex optimization, LD requires the objective function f_i to be strictly convex, whereas ADMM can be applied even when the objective function is convex. So, ADMM can be applied to wider class. Fig. 5 shows the result when $c = 0$. As readers can see, we cannot get a solution by LD for the JSP in this paper.

[1] `https://www.ims.org/`

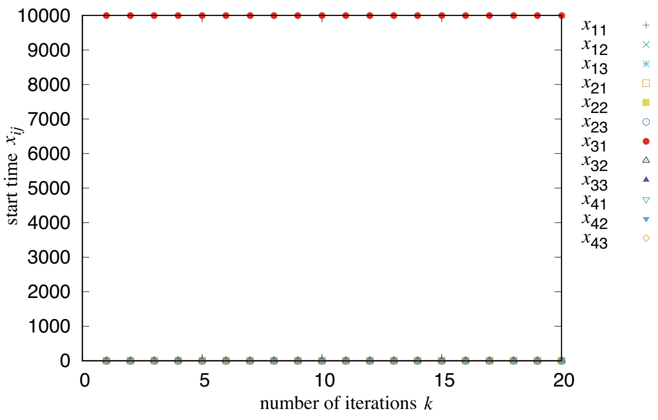

Fig. 5. Change of start time x_{ij} when LD is used and decision variables are real numbers. The algorithm has not converged and objective function value is 10000.

6. CONCLUSION

This paper proposes a method using the ADMM algorithm for the consensus problem as a distributed solution to the job shop scheduling problem. Since the ADMM algorithm for the consensus problem is a solution to the convex optimization problem, the scheduling problem, which is a non-convex optimization problem, does not guarantee optimality or convergence. The result of a computer experiment in a simple example also showed that.

In the future, we will continue to make further evaluations by applying it to larger examples, and we will investigate methods to ensure convergence and a certain degree of optimality.

REFERENCES

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Optimization*, 3(1), 1–122.

Bunce, P., Limoges, R., and Okabe, T. (1997). *NGMS — Next Generation Manufacturing Systems (IMS Project)*, 274–283. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chang, T.H., Hong, M., and Wang, X. (2015). Multi-agent distributed optimization via inexact consensus ADMM. *IEEE Transactions on Signal Processing*, 63(2), 482–497.

Ennigrou, M. and Ghédira, K. (2008). New local diversification techniques for flexible job shop scheduling problem with a multi-agent approach. *Autonomous Agents and Multi-Agent Systems*, 17(2), 270–287.

Gou, L., Hasegawa, T., Luh, P.B., Tamura, S., and Oblak, J.M. (1994). Holonic planning and scheduling for a robotic assembly testbed. In *Proceedings of the 4th International Conference on Computer Integrated Manufacturing and Automation Technology, CIMAT 1994*, 142–149. United Technologies Research Center, East Hartford, United States, IEEE Comput. Soc. Press.

Guignard, M. and Kim, S. (1987). Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical Programming*, 39(2), 215–228.

Hanada, K. and Hirayama, K. (2011). Distributed Lagrangian relaxation protocol for the over-constrained generalized mutual assignment problem.

Hayashi, N. and Takai, S. (2017). Distributed source identification by two-hop consensus dynamics with uniform time-varying communication time-delays. *SICE Journal on Control, Measurement, and System Integration*, 10(2), 70–76.

Hayashi, N., Ushio, T., and Kanazawa, T. (2010). Adaptive arbitration of fair qos based resource allocation in multi-tier computing systems. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E93-A(9), 1678–1683.

Lee, K., Leung, J.Y.T., and Pinedo, M.L. (2012). Coordination mechanisms for parallel machine scheduling. *European Journal of Operational Research*, 220(2), 305–313.

Liu, N., Abdelrahman, M.A., and Ramaswamy, S. (2007). A complete multiagent framework for robust and adaptable dynamic job shop scheduling. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, 37(5), 904–916.

Mateos, G., Bazerque, J.A., and Giannakis, G.B. (2010). Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10), 5262–5276.

Miyamoto, T., Krogh, B., and Kumagai, S. (2002). Context-dependent agents for real-time scheduling in manufacturing systems. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E85-A(11), 2407–2413.

Moore, K.L. and Lucarelli, D. (2007). Decentralized adaptive scheduling using consensus variables. *International Journal of Robust and Nonlinear Control*, 17(10-11), 921–940.

Nguyen, D.H., Narikiyo, T., and Kawanishi, M. (2018). Optimal demand response and real-time pricing by a sequential distributed consensus-based admm approach. *IEEE Transactions on Smart Grid*, 9(5), 4964–4974.

Nishi, T. and Tanaka, Y. (2012). Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 42(5), 1230–1243.

Okabe, T., Bunce, P., and Limoges, R. (1998). Next generation manufacturing systems (NGMS) in the IMS program. In *Advances in Production Management Systems*, 43–54. Springer, Boston, MA, Boston, MA.

Ren, W., Beard, R.W., and Atkins, E.M. (2005). A survey of consensus problems in multi-agent coordination. In *American Control Conference*, 1859–1864.

Segawa, K., Hamada, K., Hayashi, N., and Takai, S. (2015). Cooperative target tracking by 2-level hierarchical ptz camera sensor networks. In *IEEE Conference on Decision and Control*, 2975–2980.

Van Brussel, H., Valckenaers, P., and Wyns, J. (1997). *HMS — Holonic Manufacturing Systems Test Case (IMS Project)*, 284–292. Springer Berlin Heidelberg, Berlin, Heidelberg.

Xiong, W. and Fu, D. (2015). A new immune multi-agent system for the flexible job shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(4), 857–873.

Xu, C., Sand, G., Harjunkoski, I., and Engell, S. (2012). A new heuristic for plant-wide schedule coordination problems: The intersection coordination heuristic. *Computers and Chemical Engineering*, 42, 152–167.