Agent-Based Production Scheduling for Aircraft Manufacturing Ramp-up

Pavel Vrba[1(✉)], Ondřej Harcuba[1], Martin Klíma[2], and Vladimír Mařík[2]

[1] Czech Technical University in Prague, Prague, Czech Republic
pavel.vrba@ciirc.cvut.cz, harcuond@fel.cvut.cz
[2] Certicon a.s, Prague, Czech Republic
{martin.klima,vladimir.marik}@certicon.cz

Abstract. The paper presents a solution for scheduling of the aircraft production during the ramp-up phase. The developed scheduler features a combination of multi-agent systems and classical constraint logic programming algorithms. The concept of multi-agent systems is used to break down the complexity of the scheduling problem into smaller, independent sub-problems that can be solved independently and in parallel by individual agents. The selected use case is the Airbus A350 assembly line organized as a sequence of several assembly stations. Each station's schedule is computed by a single agent, however the coordination among agents takes place to handle situations when particular jobs cannot be finished in the original station, for instance due to missing parts, and have to be passed in the next station. The presented solution was developed within the European FP7 project ARUM.

Keywords: Multi-agent systems · Ramp-up · Aircraft manufacturing · Assembly · Scheduling

1 Context and Motivation

This paper presents one of the main outputs of the European project ARUM – Adaptive Production Management, which is carried out within the EU's Seventh Framework Programme. The ARUM project is aimed at increasing the effectiveness of production of highly complex and individualized products, such as aircrafts. The most challenging is the ramp-up period, which starts when the design of a new product is finished and the production of first products is initiated, and ends when the full capacity of the production line is reached. In case of the aircraft assembly, which is in focus of ARUM, such a period usually lasts about two to three years, but in reality may take even several extra years. An example is the grounding of fifty operating Boeing 787 airplanes in January 2013 because of the fire hazards from a new type of Lithium-Ion batteries, which stalled the production for several months [1]. Airbus A380 was also delayed in 2009 because of the design failures identified at ramp-up [2].

The reasons for lengthening of the ramp-up period are frequent disturbances that halt the assembly operations. The typical examples are missing parts caused by the

V. Mařík et al. (Eds.): HoloMAS 2015, LNAI 9266, pp. 145–156, 2015.
DOI: 10.1007/978-3-319-22867-9_13

delayed deliveries from external suppliers, non-conformant parts that are either damaged or do not conform to drawings, or late change requests from customers.

This is the case of the currently ongoing ramp-up of a new long-range aircraft Airbus A350, which represents the main test case of the ARUM project. The main goal is to develop new software tools that provide enhanced decision-making support to production managers, team leaders, and assembly line workers. The existing ERP (Enterprise Resource Planning) systems used for production planning and scheduling do not sufficiently cope with disturbances in form of provisioning the real-time updates to planned activities in order to mitigate the impacts of such disruptions. The reallocation of workers and assembly operations across the production line is done by station managers, relying mainly on their knowledge and experiences. The problem of finding the suitable work allocation is however too complex to be mastered in an optimal way, considering hundreds of depending operations in different state of completion, the varying availability of workers and materials, the currently existing disruptions like non-conformities, etc. The goal of ARUM project was thus to develop the real-time scheduling software (Scheduler in short) that supports the station managers to make relevant decisions regarding the failures mitigation. Additionally, there was a requirement to design the mobile application for workers, which keeps them updated on the assigned jobs and enables them to electronically report about the work progress as well as about any disturbances. Such detailed reporting provides a real-time feedback to the station managers as it enables them to see the actual overall status of production and helps them to decide, having the Scheduler running behind the scenes, to reorganize the work.

The job scheduling problem itself can be addressed by various means. There are mathematical optimization methods, AI-based methods, simulation-based methods, multi-agent based approaches, and others, as reviewed in Sect. 2. The approach taken in ARUM, discussed in Sect. 3, is a combination of multi-agent systems and mathematical optimization. The overall ARUM solution, described in Sect. 4, is designed according to the enterprise service bus (ESB) software architecture model, in which the loosely coupled software components – services – communicate via message exchange. One of the services is the presented Scheduler; other services are for instance the User interface service mediating the interaction with users, or the Ontology service responsible for gathering data from the legacy software systems [4]. The conclusions and prospects for future work are given is Sect. 5.

2 Related Work

In the job shop scheduling there is a set of *jobs* that must be executed in given chronological order. Each job has a given the execution time and requires particular resources such as a machine, a tool, a material, or a worker with specific skills. For given set of jobs and available resources the scheduling problem is to calculate a time table of all jobs, called *schedule*, which defines exact times at which the jobs are intended to take place and specifies which resources will be allocated to the jobs. There are multiple constraints to be satisfied such as the capacity constraints (one machine can execute only one job at the same time) or precedence constraints (jobs must be executed in given order). A solution to the scheduling problem is a schedule that satisfies all the constraints. There exist multiple solutions to the same scheduling problem, thus the goal is to find “good” solutions with respect to minimizing or maximizing given performance indicators, called *objectives*. The most common objective function is the *makespan*, which means that all the operations should be finished as soon as possible.

Production scheduling is a complex NP-hard problem, where the number of solutions grows exponentially with the problem size. There are no computationally efficient methods that find optimal solution, however the importance for manufacturing domain is indisputable. The job shop scheduling has been thus attracting researchers from different domains over the decades. As a result multitude of methods and algorithms has been designed.

A substantial part of research was devoted to *mathematical optimization algorithms*, which model the scheduling problem as mixed integer programming, solved often by the branch and bound (BB) method. BB method explores all feasible schedules by constructing a tree representation of the solution space, from which large branches of unpromising solution candidates are cut off on the basis of estimated lower and upper bounds. BB methods guarantee finding the optimal solution but due to the NP-hard nature of the scheduling problem it is applicable only on small-scale problems [3]. Instead of finding optimal solutions many approximation methods to find good near-optimal solutions were developed. It is for instance shift bottleneck procedure [4], tabu search [5], simulated annealing [6], beam search [7], genetic algorithms [8] or machine order space search [9].

In industrial practice there are more simplistic methods, such as *dispatching rules*, often used instead of elaborate mathematical algorithms. A dispatching rule tries to determine the sequence of jobs execution on a basis of some simple criteria. This can be for instance earliest due date rule that prioritizes jobs based on their due dates, or shortest process time rule that sets higher priorities for operations with shorter process times [10].

Various *artificial intelligence techniques* to scheduling problems have been developed. Examples include rule-based scheduling [11], constraint-directed search [12] and case-based reasoning [13]. A lot of attention was paid to applying expert systems technology by emulating the decision-making process of human schedulers [14]. A substantial contribution is the constraint-directed search method that uses the beam search powered reasoning engine guided by the constraints represented with schema representation language [15].

The *multi-agent system* (MAS) has been recognized as a promising conceptual framework for developing production planning and scheduling systems. Using MAS the complex system is modeled as a community of autonomous and intelligent agents that cooperate on common goals via the message-based communication. Several MAS-based methods and software prototypes for planning and scheduling have been developed, such as the methodology for real-time job-shop scheduling problems [16], the FABMAS multi-agent-system for production control of semiconductor manufacturing [17], the Production 2000+ system for scheduling and control of engine cylinder heads production [18], the ExPlanTech multi-agent planning and scheduling system deployed in Skoda Auto engine assembling workshop [19], a multi-agent system for real-time scheduling and optimization of workshop resources deployed at Axion-Holding Izhevsk factory [20], or the agent-based discrete production scheduling and control system deployed at packing line [21]. There are also several works that combine multi-agent approaches with the classical methods, such as with the dispatching rules [22], with the beam search method [23], or with the genetic algorithms [24]. A detailed review of agent-based systems for planning and scheduling can be found in [23] and [25].

3 Agent-Based Scheduler for Aircraft Manufacturing

3.1 Use-Case Description

The use-case of the ARUM project is the Airbus A350 XWB assembly line (Fig. 1). As it is usual in similar small lot productions, the line is organized as a sequence of several stations, in which the semi-finished products (in this case A350 XWB fuselages) are equipped with various components. When the assembly in a given station is completed, the product moves to the next station. This process must be synchronized across all the stations as there can be only a single product in a station at a time.

Fig. 1. Airbus A350 XWB first front fuselage in final assembly line in Toulouse, France (source: http://www.airbus.com/galleries/photo-gallery/)

The production time in the station is called a *cycle time*. At the beginning of ramp-up the actual cycle time is usually much longer than planned because of the frequent disruptions caused by late deliveries and non-conformities. As the ramp-up progresses the volatility of assembly processes settles down and consequently the cycle time shortens. The cycle time represents in fact one of the key KPIs (Key Performance Indicators) to be optimized as it has a direct correlation with the assembly line throughput.

The key challenge for the station managers is to make right decisions regarding the strategy for allocating jobs and resources to achieve a balanced production across the line with minimized impact of the actual disruptions. Because of the dependencies of jobs even a single non-conformity that caused halting a given job causes halting other dependent jobs and as a result lengthens the station's cycle time. It effects also subsequent stations in the line since they have to wait until the problem is fixed in the preceding station. It is obvious that the workers originally assigned to the impacted job have to be released and assigned to another jobs. This applies also to workers allocated to all dependent jobs, not only in the original station but also in all other stations down the line. Achieving the work balance across the whole production line therefore requires coordination of multiple station managers and thus exceeds the area of responsibility of each single one.

Besides a strategy for halting a production in a station until the problem is fixed there is another possible strategy of leaving the impacted jobs unfinished and moving them together with the product to subsequent station(s). It is for instance known that a missing part will be delivered within few days so the completion of affected jobs can be postponed to the next stations when the part arrives. This implies another challenge for station managers to coordinate their activities as the extra work caused by moving jobs can be covered either by next station's "own" workers or by borrowing the workers from the original station. In both cases a rescheduling of work across the stations has to be done again to achieve the optimal state.

3.2 Proposed Agent-Based Scheduler

The proposed scheduler features a combination of mathematical optimization methods and multi-agent systems. As seen in Fig. 2 the scheduling problem for a production line composed of Nstations is represented as $N \times N$ matrix, where lines represent the stations $S = \{S_1, S_2, \dots, S_N\}$ and columns represent the cycle times $C = \{C_1, C_2, \dots, C_N\}$.

In each cell of the matrix the scheduling task is defined as n-tuple:

$$T^{i,j} = \{J^{i,j}, D^{i,j}, SK^{i,j}, R^{i,j}, O^{i,j}\}, i = 1, \dots, N; j = 1, \dots, N$$

where:

- $J^{i,j} = \left\{J_1^{i,j}, \dots, J_{M^{i,j}}^{i,j}\right\}$ is a set of $M^{i,j}$ jobs to be performed on a product; each job has a given duration $dur\left(J_x^{i,j}\right)$;
- $D^{i,j}$ is a set of job dependencies represented as doubles $\{J_a, J_b\}, J_a, J_b \in J^{i,j}; a \neq b$ for which holds $endtime(J_a) \leq starttime(J_b)$;

- $SK^{i,j}$ is a set of skills required to perform particular jobs; $SK^{i,j} = \left\{SK_1^{i,j}, \dots, SK_{M^{i,j}}^{i,j}\right\}$, where $SK_x^{i,j}$ is a set of skills required by job $J_x^{i,j} \in J^{i,j}$
- $R^{i,j} = \left\{R_1^{i,j}, \dots, R_{M^{i,j}}^{i,j}\right\}$ is a set of available resources, where $R_x^{i,j}$ is a set of the skills provided by the x-th resource;
- $O^{i,j}$ is the objective function to assess the quality of solutions.

The goal of scheduling is for each job $J_x^{i,j} \in J^{i,j}$ to compute its start time $starttime\left(J_x^{i,j}\right)$ and end time $endtime\left(J_x^{i,j}\right)$, such as the constraints on durations of the jobs expressed as $starttime\left(J_x^{i,j}\right) + dur\left(J_x^{i,j}\right) = endtime\left(J_x^{i,j}\right)$ are met and there are resources with required skills available during the job's execution times. Some of the resources might be modeled as *cumulative constraints*, meaning that there is k units of the resource available at the same time, while the given job consumes $l, l \leq k$ units of the resource.

In detail, Fig. 2 depicts a case of a beginning of the current cycle C_1, with the product P_N that has just entered the first station of the line, while other products $P_{N-1}, \dots, P_1$ were moved one station ahead. At the same time the product P_0 (not shown in Fig. 2) left the last station S_N. In the next cycle C_2 the same scenario repeats – P_{N+1} enters the line, P_1 leaves it and all other products move ahead (shown as diagonal movement in the matrix).

If only the mathematical optimization algorithms would be used to solve the scheduling problem then the model would have to be built from the complete matrix $T = \{T^{i,j}\}, i = 1, \dots, N; j = 1, \dots, N$ because of the interrelations between particular matrix cells, especially due to unfinished jobs allowed to move across stations. Considering the NP-hard character of the problem this might imply serious issues regarding the unrealistic computation times.

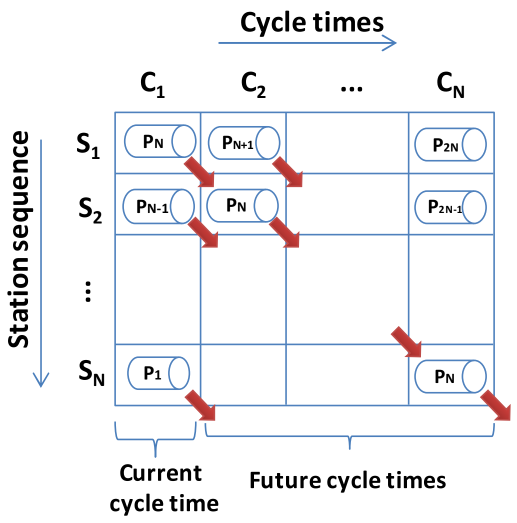

Fig. 2. The graphical representation of the scheduling problem, considering N stations and a scheduling horizon given by N cycle times

To reduce the complexity the mathematical optimization methods were combined with the multi-agent systems concept. In this approach each scheduling task $T^{i,j}$ is solved independently by an agent and, if needed, the coordination between the agents takes place to deal with the traveling jobs. The *Station agent* responsible for computation of task $T^{i,j}$ uses the constraint logic programming engine – the open-source Choco solver (http://choco-solver.org/). The model for the solver includes variables, representing the start times and end times of jobs and their durations. The constraints are expressed as mathematical functions of the variables, such as `job_starttime + job_duration = job_endtime` or `job1_endtime ≤ job2_starttime`. The solver explores the state space and searches for such values of variables so that all the constraints are met. Different heuristics are applied to limit the searched state space, such as placing the jobs with higher amount of dependencies first, etc. (detailed discussion is out of scope of this paper).

The solver generates multiple valid solutions, which are evaluated using the objective function $O^{i,j}$ to find the "good" ones. Two different objective functions are considered:

1) $O^{i,j} = min\left\{max\left\{endtime\left(J_x^{i,j}\right)\right\}\right\}$, $\forall J_x^{i,j} \in J^{i,j}$, which is translated to finding such a configuration in which the last job is finished as early as possible. It is in fact equal to minimizing the *make span*. The main motivation is to create an "empty space" at the end of the cycle time, as shown in Fig. 3. There are two reasons for this: (i) to have a reserve for possibly delayed jobs in the same station, and (ii) to be able to accommodate traveling jobs from previous station.
2) $O^{i,j} = min\left\{\sum_{J_x^{i,j}} f\left(starttime\left(J_x^{i,j}\right), dur\left(J_x^{i,j}\right)\right)\right\}$, which purpose is to find such a configuration in which all the jobs are finished as early as possible. The idea is to divide the time interval $\langle starttime\left(J_x^{i,j}\right), endtime\left(J_x^{i,j}\right)\rangle$ to pieces of the length 1 and give the penalty to each of that piece equal to the absolute position of the piece in time from the beginning of the cycle time. If, for example, $starttime\left(J_x^{i,j}\right) = 10$ and the job has duration equal to 3 time units, there are three pieces of length 1 placed at positions 10, 11, and 12, penalized by same values 10, 11, and 12, respectively. It gives the total penalty equal to 33. It is in fact a sum of integer numbers in a given interval, the function $f(\cdot)$ from the formulae above is thus:

$$f(\cdot) = \frac{\left(starttime\left(J_x^{i,j}\right) + \left(starttime\left(J_x^{i,j}\right) + dur\left(J_x^{i,j}\right) - 1\right)\right) \cdot dur\left(J_x^{i,j}\right)}{2}$$

The decomposition of work among the Station agents is done by the *Scheduler agent*. Its task is to obtain the complete data set for the scheduling matrix, decompose it into sub-problems $T^{i,j}$ and orchestrate the computation done by Station agents. First, the Station agents in the first column of the matrix compute their results and return them back to the Scheduler agent, marking the jobs that cannot be suitably placed, for instance due to the missing resource needed for the execution. The Scheduler agent analyzes the results and makes a decision about the failed jobs. If the strategy is to halt the production until the problem is fixed then the cycle time is extended according to the expected date and time of the fix ([S_i, C_{j+1}] in Fig. 3). In case of traveling work, the affected jobs are left marked and are then handled in the next step.

In this step the Station agents for the next column in the matrix are requested to compute their schedules, giving them their "own" jobs plus the additional traveling jobs from the previous station (see the highlighted job moving from [S_i, C_j] to [S_{i+1}, C_{j+1}] in Fig. 3). Handling the extra jobs follows another two strategies: (i) using the target station's own resources ([S_{i+1}, C_{j+1}]), or (ii) borrowing the resources from the original station. In the latter case however, it does not mean to borrow from [S_i, C_j], but in fact from [S_i, C_{j+1}] to be in the right cycle time. This situations is handled in such a way that the two involved agents agree on choosing one of them as a master, which then combines data, including the jobs and resources, from both $T^{i,j+1}$ and $T^{i+1,j+1}$ cells and lets its solver to compute the result. The result is then split by the master agent back to two groups according to the original belonging of the jobs to stations. However, the result contains the information about some resources (mainly the humans) that will be requested to move from one station to the other one to work on the traveling jobs.

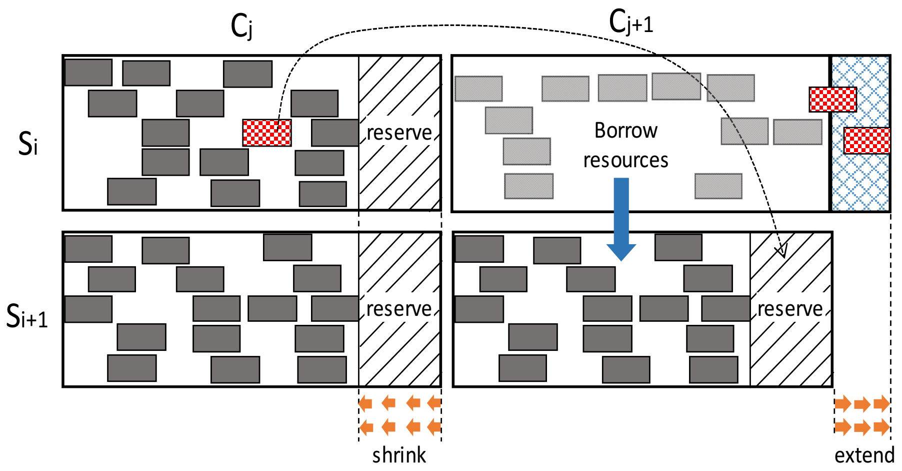

Fig. 3. Possible scheduling results – shrinking or extending the cycle time; moving jobs between stations with borrowing resources

This procedure is repeated until the computation for the last cycle time is finished. Finally, the overall solution for the whole matrix is assembled by the Scheduler agent, which then computes the global KPIs (key performance indicators). The KPIs are presented to the station manager as a quality of found solution. It is the number of traveling jobs, the human utilization and the station flexibility representing the ratio of the reserve at the end of cycle time to the cycle time length (see Fig. 3).

As the production progresses and the disruptive events are occurring the station manager can decide to run the re-scheduling process. The goal is to find a solution that is as close to the previous one as possible. It is mainly because the just-in-time delivery of parts that would be harmed if the jobs would be shifted in time too much. For this reason the Station agents use the last found solution as a starting point for the solver to begin searching the state space.

It is shown in Fig. 3 that besides extending the cycle time it is also possible to decide shrinking it, if the time reserve is achieved in all the stations in the same cycle time. This is one of the main contributions of the presented soultion as it results in speeding up the production and achieve higher throughput of the line.

4 Integration of the Scheduler into the Overall ARUM Solution

The ARUM system is designed according to the Enterprise Service Bus (ESB) software architecture model, which is one of the implementations of the Service Oriented Architecture (SOA) paradigm. The software modules are designed as self-contained loosely-coupled services that communicate via messages over the messaging bus.

Fig. 4 shows services that are involved in the scheduling task. It is naturally the *Scheduler Service* implemented as a multi-agent system. It includes a MAS-ESB gateway, which was designed to enable the agents, implemented in JADE agent platform (http://jade.tilab.com/) to interact with the JBoss ESB (http://jbossesb.jboss.org) services [26].

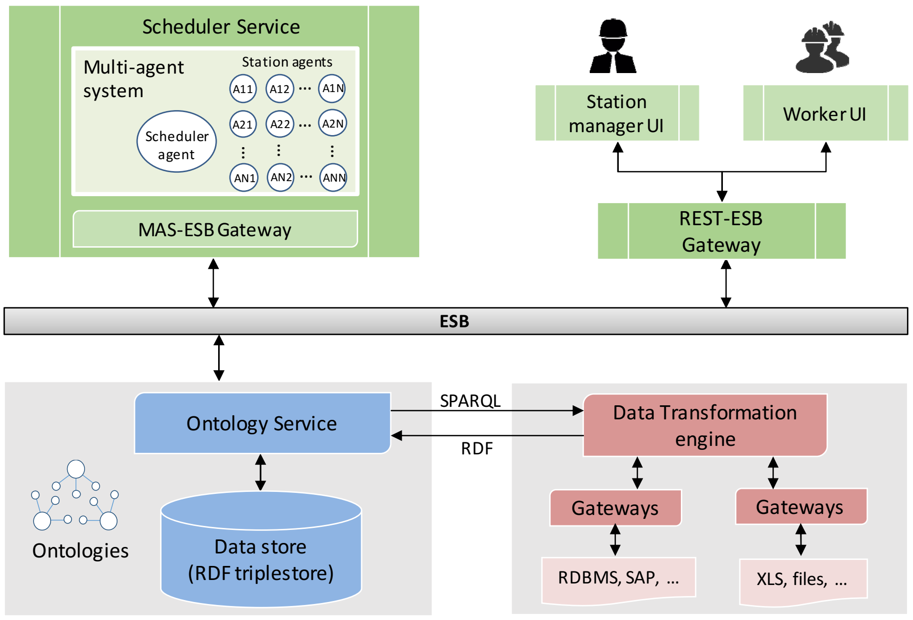

Fig. 4. Integration of the Scheduler (Scheduler Service) with the Ontology Service providing it with data and the user interfaces from which the scheduler is controlled

The *REST API Service* provides means for integration of lightweight clients with the heavyweight Enterprise Service Bus (ESB) messaging system via the HTTP-based RESTful API [27]. These clients are primarily the user interfaces for both the station managers and workers. The *Station manager UI* provides the station manager with the real-time process monitoring and control. It gives a detailed overview of the progress of work in the stations, such as a number of finished/unfinished jobs, estimated lead time, current throughput, utilization of resources, occurred events, etc. Through this user interface the station manager creates a new schedule at the beginning of the cycle time as well as at the beginning of each work shift. He/she can also decide to update the schedule any time when coming to a conclusion that the events such as missing parts and non-conformities disrupt the currently running production to such extent it is necessary to re-organize the production and re-allocate the workers. The Station manager UI interacts over the REST-API Service with the Scheduler Service requesting it to compute a new or update an existing schedule. When finished, the results are displayed to the station manager, who decides to apply them to a running production.

The new allocation of work is propagated down to workers that are equipped with a smart phone/tablet running the *Worker UI*. Through this mobile application the worker checks the list of jobs he is supposed to execute during the work shift, reports on the progress of work (starting and completing the job) and also report on any disturbances such as missing/broken parts or other non-conformities.

The *Ontology Service* is responsible for gathering data from the legacy software systems and converting them into a common data format. It was decided to use ontologies to model the data structures in ARUM in a semantic, machine interpretable form. The Ontology Service queries the Data Transformation engine, which gathers data over the Gateways from various legacy systems, such as relational databases, SAP, Microsoft Excell spread sheets, etc. Then data is converted into RDF (Resource Description Framework) [28] according to an ontology developed for modeling of discrete production processes. The Ontology Service aggregates data from multiple sources and stores them into an RDF triplestore. In this way the input data for the Scheduler Service are prepared. When the scheduling is finished, the output of the scheduling (the assignment of resources to jobs, and job's planned start and end times) are also stored in a semantic form by the Scheduler Service. Subsequently, the results are displayed in the Station manager UI, which involves conversion of data in RDF format handled by the Ontology Service into the JSON format appropriate for lightweight clients.

5 Conclusions and Future Work

The presented paper reports on the scheduling system developed to increase the efficiency of the ramp-up production of Airbus aircrafts. The solution features a combination of constraint logic programming and the multi-agent systems, which helps to break down the complexity of the scheduling problem. The experiments show that if only the constraint programming would be used, with the model containing data for the whole matrix (Fig. 2), the solution is not found by the solver even within a couple of hours. Using the agents finding a good enough solution takes couple of minutes; of course due to the NP-hard character of the problem, it is not the globally optimal one.

We are currently preparing experiments that should give the estimates of improvement of the production performance when the presented solution is deployed in the real factory. The experiments will employ an emulator of the disruptive events that uses a statistical data of the real occurrence of failures gathered during the ongoing A350 ramp-up. The goal is to compare the real performance with a state when the continuous rescheduling would be applied. The expectations are quite high, going up to 30%.

The presented solution obviously gives better results than the manual ad-hoc re-scheduling done by the station managers, for it considers the interrelations between the particular stations instead of relying on the local view of individual station managers only. In addition to the presented use case, the Scheduler is supposed to be used also in a combination with the strategic planner to validate different possible configurations of the production line. The attention is paid to so-called "parking station", where the product, which production is halted due to a disruptive event, can be put aside and wait until the problem is fixed. In this way the flow in the main line would be not affected. The challenge here is to determine a proper moment when to put the product from the parking station back to the line, in which all the stations are occupied by other products. It requires to speed up the production in part of the line and simultaneously slow down the progress in the preceding part of line to vacate a given station in between, to which the product from the parking place will be returned. This again requires a coordination among the Station agents in terms of negotiating about reallocating people from the between the parts of the line to achieve a requested misbalance.

Acknowledgements. This research was supported by the European Union FP7 Programme under the ARUM project No. 314056, and also by the Ministry of Education, Youth and Sports of the Czech Republic under the grant No. 7E13001.

References

1. Dreamliner Grounded Until At Least End Of May. In: Sky News, February 25, 2013 (accessed June 12th 2015). http://news.sky.com/story/1056514/dreamliner-grounded-until-at-least-end-of-may
2. Kingsley, J.M.: Airbus slows A380 final assembly ramp-up. In: Flightglobal, May 14, 2009 (access February 25th 2013). www.flightglobal.com/news/articles/airbus-slows-a380-final-assembly-ramp-up-326416/
3. Jain, A.S., Meeran, S.: Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research **113**(2), 390–434 (1999)
4. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job shop scheduling. Management Science **34**(3), 391–401 (1988)
5. Amico, M.D., Trubian, M.: Applying tabu search to the job-shop scheduling problem. Annals of Operations Research **41**(3), 231–252 (1993)
6. Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K.: Job shop scheduling by simulated annealing. Operations Research **40**(1), 113–125 (1992)
7. Ow, P.S., Morton, T.E.: Filtered beam search in scheduling. International Journal of Production Research **26**(1), 35–62 (1988)
8. Jensen, M.T.: Generating robust and flexible job shop schedules using genetic algorithms. IEEE Transactions on Evolutionary Computation **7**(3), 275–288 (2003)
9. Choi, S.H., Yang, F.Y.: A machine-order search space for job-shop scheduling problems. International Transactions in Operational Research **10**(6), 597–610 (2003)
10. Pinedo, M.: Scheduling: Theory, algorithms, and systems. 4th edn. Springer Verlag (2012)

11. Bruno, G., Elia, A., Laface, P.: A rule-based system to schedule production. IEEE Computer **19**(7), 32–40 (1986)
12. Fox, M.S., Smith, S.F.: ISIS-a knowledge-based system for factory scheduling. Expert systems **1**(1), 25–49 (1984)
13. Miyashita, K., Sycara, K.: CABINS: a frarnework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. Artificial Intelligence **76**(1–2), 377–426 (1995)
14. Metaxiotis, K.S., Askounis, D., Psarras, J.: Expert systems in production planning and scheduling: a state-of-the-art survey. Journal of Intelligent Manufacturing **13**, 53–260 (2002)
15. Fox, M.S.: Constraint-directed search: A case study of job-shop scheduling, Ph.D. Thesis, Carnegie Mellon University, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh, PA, CMU-RI-TR-85-7 (1983)
16. Choi, S.H., Yang, F.Y.: A machine-order search space for job-shop scheduling problems. International Transactions in Operational **10**(6), 597–610 (2003)
17. Mönch, L., Stehli, M., Zimmermann, J., Habenicht, I.: The FABMAS multi-agent system prototype for production control of water fabs: design, implementation and performance assessment. Production Planning & Control **17**(7), 701–716 (2006)
18. Bussmann, S., Schild, K.: An agent-based approach to the control of flexible production systems. In: 8th IEEE International Conference on Emerging Technologies and Factory Automation, vol. 2, pp. 481–488 (2001)
19. Pěchouček, M., Říha, A., Vokřínek, J., Mařík, V., Pražma, V.: ExPlanTech: Applying Multi-agent Systems in Production Planning. International Journal of Production Research **40**(15), 3681–3692 (2002)
20. Andreev, M., Ivaschenko. A., Skobelev, P., Tsarev, A.: Multi-agent platform design for adaptive networks of intelligent production schedulers. In: 10th International IFAC Workshop on Intelligent Manufacturing Systems, pp. 78–83 (2010)
21. Vrba, P., Radakovič, M., Obitko, M., Mařík, V.: Semantic extension of agent-based control: the packing cell case study. In: Mařík, V., Strasser, T., Zoitl, A. (eds.) HoloMAS 2009. LNCS, vol. 5696, pp. 47–60. Springer, Heidelberg (2009)
22. Merdan, M., Moser, T., Sunindyo, W., Biffl, S., Vrba, P.: Workflow scheduling using multi-agent systems in a dynamically changing environment. Journal of Simulation **7**(3), 144–158 (2012)
23. Gang, K.S., Hong, C.: Multi-Agent Based Beam Search for Real-Time Production Scheduling and Control: Method, Software and Industrial Application. Springer, London Heidelberg New York Dordrecht (2013)
24. Shen, W.: Implementation of genetic algorithms in agent-based manufacturing scheduling systems. Integrated Computer-Aided Engineering **9**(3), 207–218 (2002)
25. Shen, W., Hao, Q., Yoon, H.J., Norrie, D.H.: Applications of agent-based systems in intelligent manufacturing: an updated review. Advanced Engineering Informatics **20**(4), 415–431 (2006)
26. Vrba, P., Fuksa, M. Klíma, M.: JADE-JBossESB gateway: integration of multi-agent system with enterprise service bus. In: IEEE International conference on System, Man, and Cybernetics, pp. 3663–3668 (2014)
27. Harcuba, O., Vrba, P.: Unified REST API for supporting the semantic integration in the ESB-based architecture. In Proceedings of IEEE International Conference on Industrial Technology (2015)
28. Manola, F., Miller, E., McBridge, M.: RDF 1.1 primer (2014) (accessed June 12, 2015). http://www.w3.org/TR/2004/REC-rdf-primer/20040210