This is the author's version of an article that has been published in the CASE 2018 proceedings.
Changes were made to this version by the publisher prior to publication.
The final version of record is available at https://dx.doi.org/10.1109/COASE.2018.8560544

Specifying and Synthesizing Energy-Efficient Production System Controllers that Exploit Braking Energy Recuperation

Daniel Gritzner[1], Elias Knöchelmann[2], Joel Greenyer[1], Kai Eggers[2], Svenja Tappe[2] and Tobias Ortmaier[2]

***Abstract*— Reducing the energy consumption is a major concern in industrial production systems. One approach is recuperating the braking energy of robot axes. Ideally, their acceleration and deceleration phases should be synchronized so that the braking energy of one axis can be reused directly to accelerate another. This requires a detailed alignment of the axes' trajectories, but also a careful design of the overall discrete control. Finding an optimal control strategy manually, however, is difficult, as also many functional and safety requirements must be considered. We therefore propose an automated methodology that consists of three parts: (1) A scenario-based language to flexibly specify the discrete production system behavior, (2) an automated procedure to synthesize optimal control strategies from such specifications, including PLC code generation, and (3) a procedure for the detailed trajectory optimization. We describe the methodology, focusing on parts (1) and (2) in this paper, and present tool support and evaluation results.**

I. Introduction

Due to increased automation, it becomes more and more important to optimize the energy consumption of production processes. This can be done, for example, by reducing idle times [1]–[3]. For production robots, it is possible to optimize the axes' trajectories for energy-efficiency [4]–[6]. Even more energy saving potential lies in reusing the braking energy of robot axes to accelerate other robot axes in the same system [7], [8], instead of allowing the braking energy to dissipate as heat, which is the usual practice today.

Recuperated braking energy can be exchanged between robots via a common DC link. The challenge is, however, to optimally synchronize the braking and acceleration phases of robot axes to exploit the energy saving potential.

Optimally synchronizing the braking and acceleration phases of robot axes is a challenge on two levels: First, the overall discrete control of the system must be optimized to maximize the times in which the acceleration and deceleration phase of axes beneficially overlap. Usually, the order of robot movements can be arranged in multiple ways, and some control strategies offer more braking energy recuperation potential than others. Then, for each set of braking and acceleration phases that were brought to overlap by an optimized control strategy, the trajectories of the involved axes must be aligned in detail. For example, the braking phase of one axis can be shifted, stretched, or compressed to match the energy demand for accelerating another axis. We call these two levels the *sequential control optimization (SCO)* and *detailed trajectory optimization (DTO)*. In this paper, we focus on the SCO.

The SCO can be a complex task, as it could imply rethinking the entire discrete control strategy and rewriting hundreds or thousands of lines of PLC code. If, during DTO, it turns out that a particular set of braking and acceleration phases cannot be aligned as beneficially as expected during a first SCO, even iterations of SCO/DTO steps may be necessary. Doing this manually is very difficult and time-consuming, as also many functional and safety-critical requirements must be considered.

We therefore propose a methodology for specifying and automatically synthesizing controllers for production systems that reduce energy consumption by exploiting braking energy recuperation. The process is shown in Fig. 1; it automates the SCO/DTO steps and mainly consists of three parts:

(1) Scenario-based specification (① in Fig. 1): The methodology offers engineers a flexible language for specifying the discrete control behavior of a production system. This language, called the *Scenario Modeling Language (SML)* [9] is a scenario-based language that allows engineer to specify, in a set of separate stories, how the system may, must, or must not react to particular events. Assumptions on the behavior of the physical/mechanical parts of the system can also be specified, as well as assumptions on the energy quantities consumed and produced by robot arm acceleration resp. deceleration phases.

This approach has several advantages. First, the scenarios in a specification are loosely coupled; each scenario can extend as well as restrict certain aspects of the system's behavior. Using this paradigm, in contrast to programming sequential PLC code, does not encourage the engineer to introduce an order among the events in the system where such an order is not strictly dictated by the requirements. As a consequence, this naturally leaves a larger space of control strategy choices that can be exploited for SCO later on. Second, the scenarios are *executable*, which means that the engineer can simulate and validate the behavior that unfolds from the interplay of the scenarios. Third, the scenarios have a formal semantics, and can be used for formal analysis, e.g., to find inconsistencies in the specification, or for formal controller synthesis, i.e. the automatic construction of a state-based model that is a correct-by-construction implementation of the specification. This brings us to the second component of our methodology.

*This research is funded by the DFG project EffiSynth.

[1]Daniel Gritzner and Joel Greenyer are with the Software Engineering Group, Leibniz Universität Hannover, 30167 Hanover, Germany {daniel.gritzner|greenyer}@inf.uni-hannover.de

[2]Elias Knöchelmann, Kai Eggers, Svenja Tappe, and Tobias Ortmaier are with the Institute of Mechatronic Systems, Leibniz Universität Hannover, 30167 Hanover, Germany, elias.knoechelmann@imes.uni-hannover.de

This is the author's version of an article that has been published in the CASE 2018 proceedings.
Changes were made to this version by the publisher prior to publication.
The final version of record is available at https://dx.doi.org/10.1109/COASE.2018.8560544

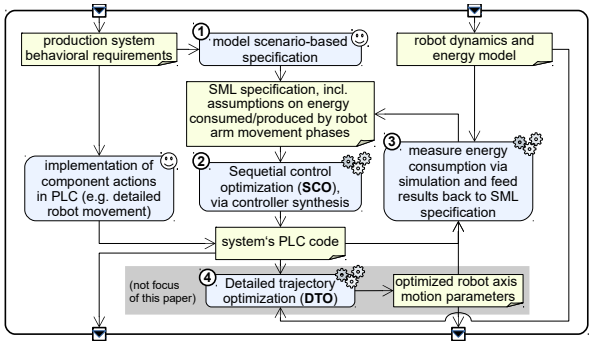

Fig. 1. Overview of our proposed engineering methodology.

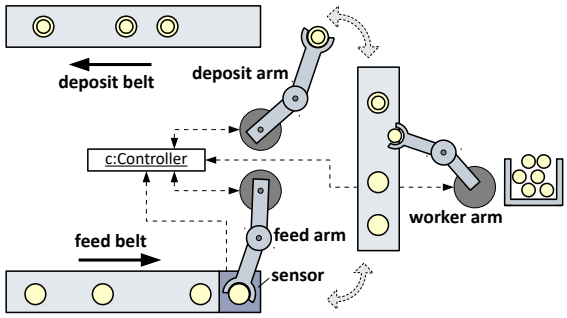

Fig. 2. Example of a production cell with three robot arms and three conveyor belts. Two arms transport the work items between the conveyor belts while the third arm modifies the work items, e.g., by adding screws.

(2) Automated SCO via Energy-Optimal Controller Synthesis (② in Fig. 1): All possible executions of the scenarios in a specification can be captured in a graph where the edges are labeled with *environment events*, representing sensor inputs, and *system events*, representing output signals to actuators, e.g. to move a robot arm. This graph can be seen as a two-player game between the system and its environment where the objective of the system is, first, to choose system steps in a way that the specification is satisfied no matter what steps the environment takes. Second, the system should choose steps such that the resulting execution consumes minimal energy.

In order to compute a strategy for this objective, we applied an algorithm for computing strategies in infinite games [10], and extended it to determine the most energy-efficient strategy [11]. This extension is based on a function to determine the overall energy cost of path fragments where energy-consuming processes take place and possibly overlap with energy-producing processes, for example, a path fragment between starting and stopping the acceleration of robot *A* that may overlap with starting and stopping the deceleration of robot *B*. Based on this function, we determine which strategy has minimal energy cost per processed work item.

The resulting strategy can be compiled to PLC code [12]. Combined with manually implemented PLC components that control the detailed robot arm movement actions (cf. [12]), the resulting PLC code can be executed to drive the final system or to simulate the system. With a detailed dynamics model of the robot arms, including energy consumption/production profiles, a detailed assessment of the energy consumption of the production process is possible via simulation (③ in Fig. 1).

(3) Trajectory optimization (④ in Fig. 1): From the synthesized control strategy, it is possible to derive which robot arm's acceleration and deceleration phases overlap, which is the input for DTO. Based on the DTO results, the SCO/DTO process can be carried out iteratively.

In this paper we focus on explaining the SCO part of the proposed methodology. The DTO technique is already worked out [13], but the integration of SCO and DTO is still ongoing work. The novel contribution of this paper is that we introduce scenario-based specifications and formal controller synthesis to production systems in a way that controllers that maximize energy recuperation can be computed automatically.

We illustrate the methodology using an example of a production cell presented in Sect. II. In Sect. III, we describe the specification approach and synthesis procedure. In Sect. IV we describe the energy model, which allows us to simulate the overall energy consumption of a production process. We report on evaluation results in Sect. V, discuss related work in Sect. VI, and conclude in Sect. VII.

II. Example

As a running example we use a production cell as illustrated in Fig. 2. Our example consists of three robot arms and three conveyor belts. Work items arrive via the feed belt and, after they have been detected by a sensor, they are transported to the next belt by the feed arm. A worker robot then modifies the work items by adding screws. Finally, the deposit arm transports the work items to the deposit belt. This process is driven by a single controller, which receives all sensor events and sends appropriate instructions to all actuators.

III. Scenario-based Specification and Synthesis

The Scenario Modeling Language (SML) [9] is a text-based variant of Life Sequence Charts [14]. With SML, a formal behavior specifications can be modeled as collections of short scenarios. An SML specification partitions the components of the system into *controllable* and *uncontrollable* components. Controllable components are usually software controllers (the central controller in our example). Uncontrollable components are the controlled physical/mechanical parts of the system (the other components in our example).

The scenarios are either *guarantee scenarios* or *assumption scenarios*. Guarantee scenarios describe how software controllers may, must, or must not react to events, and *assumption scenarios* describe what may, will, or will not happen in the controlled system, or how the controlled system, in turn, reacts to controller signals. Listing 1 shows an excerpt of an SML specification which models the process shown in Fig. 2.

```
1  [...]
2  guarantee scenario FeedRobotDeliversItem {
3     feedRobot -> controller.pickedUpItem()
4     eventually controller -> feedRobot.moveTo(workBelt)
5     feedRobot -> controller.arrivedAt(workBelt)
6     wait eventually [workerRobot.location
7           == workerRobot.idleLocation]
8     urgent controller -> feedRobot.releaseItem()
9     feedRobot -> controller.releasedItem()
10    eventually controller -> feedRobot.moveTo(feedBelt)
11 }
12 [...]
13 assumption scenario ArmMovesToLocation {
14    controller -> robot.moveTo(bind targetLocation)
15    committed robot -> controller.accelerate()
16    eventually robot -> controller.movingAtConstantVelocity()
17    eventually robot -> controller.decelerate()
18    eventually robot -> controller.arrivedAt(targetLocation)
19 }
20 assumption scenario FeedRobot_Acceleration cost [1.0] {
21    feedRobot -> controller.accelerate()
22    feedRobot -> controller.movingAtConstantVelocity()
23 }
24 [...]
```

Listing 1. Excerpt of a SML specification for our example cell

We assume that an execution of a system is an infinite sequence of *message events*. A message event is a synchronous messages sent between two components. A message event sent from an uncontrollable component is an *uncontrollable event*, for example a signal sent from a robot or a sensor to the controller (e.g., lines 3, 5, 9). A message event sent from a controllable component is a *controllable event*, for example a signal sent from the controller to a robot (e.g., lines 4, 8, 10).

Each scenario formulates rules over sequences of message events, and, by interweaving the scenarios based on common events (e.g. line 4 and 14), they yield a coherent system behavior model. The system components are represented in the scenarios by *roles* (e.g., controller and robot). Roles can either refer to a specific component, such as controller or feedRobot, or to any component of a certain type, such as robot. The latter means that a scenario applies for multiple components that share the specified aspects of behavior.

The guarantee scenario FeedRobotDeliversItem, for example, describes how the the feed robot shall deliver a work item: after the feed robot picks up an item, the controller commands the robot to move to the work belt. When the robot arrives there and the worker robot is idle, i.e., ready to process the next item, the feed robot releases the item. After it has done so, the controller instructs the robot to move back to its original position. The keyword *eventually* marks message events that the scenario requires to occur at some point in the future. The keyword *urgent* says that the event is required to occur before the next environment event occurs.

The assumption scenario ArmMovesToLocation models how we assume that robot arm movements take place: when a robot is instructed to move to a location, it will immediately start accelerating. The keyword *committed* means that the event should occur immediately, and before any other event takes place. After accelerating, the robot will eventually reach a constant velocity, then eventually starts decelerating, and then eventually it arrives at its destination.

Line 20 in Listing 1 shows how we model the energy consumption of an action. The example shows an annotation of a scenario modeling the acceleration phase of a robot's movement. The cost annotation indicates that the assumed energy consumption of this movement phase is 1 unit. In later iterations of our engineering process (Fig. 1) this value is replaced by measured energy values. The annotated values may be negative to indicate that braking energy is generated.

The interweaving of the scenarios can be done via the *play-out* algorithm [9], [14]. Play-out describes how the scenarios are executed in reaction to a sequence of uncontrollable events. We refer to [9] for details. Play-out can often make many non-deterministic choices to execute different events requested by the different scenarios. When events are annotated with the eventually modality, play-out can also choose to delay the execution of an event.

All possible play-out executions for a system can be captured in a so-called *play-out graph*. It contains all the different choices of the play-out algorithm to pick events or to delay them. An excerpt of the play-out graph of our example specification is shown in Fig. 3. Each state in this graph is defined by the state of all components in the system and a set of active scenarios in their current state of progress. Edges labeled with uncontrollable events have a dashed line, and a solid line when labeled with a controllable event. States have either only outgoing edges labeled with controllable events (including a wait event), or they have only outgoing edges labeled with uncontrollable events. The states are then also called *controllable* resp. *uncontrollable*.

To generate executable code from an SML specification we must first compute a deterministic strategy of how to select events in controllable states so that, no matter what choices are made in uncontrollable states, the resulting execution fulfills all guarantee scenarios or violates at least one assumption scenario. Scenarios can be *violated* either by choosing an event that the scenario forbids, or by not sending an event that the scenario requires. This strategy computation is called *controller synthesis*.

We adopted a game solving algorithm [10] to do this for SML specifications.

From a resulting strategy, PCL code can be generated [12].

We extended the synthesis algorithm to extract a strategy where the energy cost per item processed is minimal [11]. This extension is based on a function to determine the overall energy cost of an execution sequence, which is the energy consumed by all the components minus the recuperated braking energy. Given a sequence of events on a path in the play-out graph, we can deduce which acceleration and braking phases overlap, which then allows us to sum up the energy values annotated on the scenarios that are progressed along that path. This calculation makes the assumption that the overlapping acceleration and braking phases can always be aligned so that all braking energy is recuperated, but this may not be the case in reality. Thus, the consumption of a control strategy must always be determined by a simulation that takes into account the detailed robot dynamics and their energy model. Based on the simulation results, the energy value annotations on the scenarios can be updated and SCO process repeated.

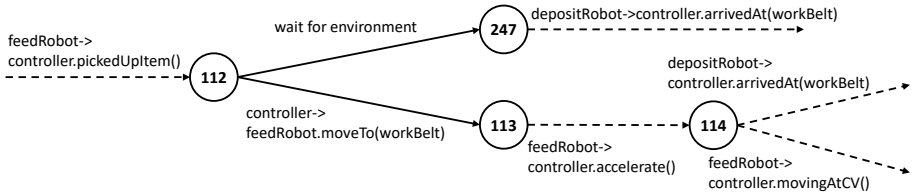

Fig. 3. Excerpt of a state graph induced by Listing 1. The nodes represent states and the labeled edges represent events. Solid arrows indicate that an event is a controllable system event, whereas dashed arrows indicate than an event is an environment event.

IV. Detailed Power Consumption Model

This section describes the detailed power consumption model for the DTO and measurement of energy values via simulation. The model provides motion-dependent power and energy consumption for each component. The following equations depict the power consumption model for a single industrial robot. Assuming that the trajectory is predefined by the task, the calculation starts with determining the vector of motor torques $\boldsymbol{\tau}(t)$. In general, the model of inverse dynamics is given by

$$\boldsymbol{\tau}(t) = \mathrm{diag}\left(\frac{1}{u_{\mathrm{G},1}}, \ldots, \frac{1}{u_{\mathrm{G},n}}\right)\left(\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})\right) + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \tag{1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are time-dependent joint angles, velocities, and accelerations given by the trajectory planning algorithm. The term $u_{\mathrm{G},i}$ represents the gear factor of joint i while vector $\boldsymbol{\tau}$ contains the respective motor torques τ_i. $\boldsymbol{M}$ contains moments of inertia, $\boldsymbol{c}$ Coriolis effects, and $\boldsymbol{g}$ gravitational effects. $\boldsymbol{h}$ summarizes non-linear effects which in our regarded case is merely friction. In [15], a commonly used friction model including Coulomb friction and viscous damping (coefficients $f_{\mathrm{c},i}$ and $f_{\mathrm{v},i}$, respectively) is presented. It is applied to this model, expressing friction torque $\tau_{\mathrm{f},i}$ of joint i as

$$\tau_{\mathrm{f},i}(t) = h_i(t) = f_{\mathrm{c},i}\ \mathrm{sign}(\dot{\varphi}_i(t)) + f_{\mathrm{v},i}\ \dot{\varphi}_i(t), \tag{2}$$

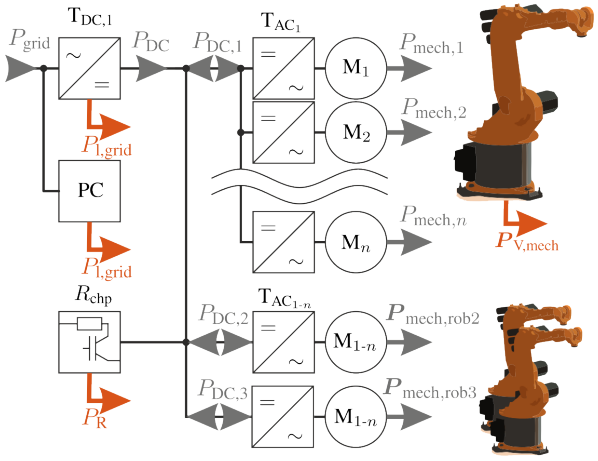

Fig. 4. Electrical substitute circuit diagram for an industrial robot and the DC bus.

where $\dot{\varphi}_i$ is the angular motor velocity of motor i which can be determined as

$$\dot{\varphi}_i(t) = u_{\mathrm{G},i}\ \dot{q}_i(t). \tag{3}$$

Most robotic manufacturers utilize an inverse dynamics model within the robot control system for implementation of feed forward control. Thus, it can be assumed that the system friction parameters are known. If not, they can be obtained using established identification methods [16], [17]. Equations (1) and (3) are used to obtain the mechanical power $P_{\mathrm{mech},i}(t)$ of each motor i:

$$P_{\mathrm{mech},i}(t) = \tau_i(t)\ \dot{\varphi}_i(t). \tag{4}$$

The DC bus power P_{DC} that describes the power flow between DC bus and robot (see Fig. 4) is obtained by summing up the mechanical power of the n individual motors:

$$P_{\mathrm{DC,Robot}}(t) = \sum_{i=1}^{n} P_{\mathrm{mech},i}(t). \tag{5}$$

The calculation is repeated for every DC bus participant $j = 1..m$. The resulting DC bus powers $P_{\mathrm{DC},j}$ are summed up to gain total DC bus power P_{DC}:

$$P_{\mathrm{DC}}(t) = \sum_{j=1}^{m} P_{\mathrm{DC},j}(t). \tag{6}$$

Further, the rectifiers in industrial robot cabinets are usually not able to recuperate. Hence, negative values of P_{DC} need to be partly corrected. Excess power in generator operating phases can cover constant losses $P_{\ell,\mathrm{DC}}$ within the DC bus, but grid side losses $P_{\ell,\mathrm{grid}}$ (such as controller, cooling fans, actively lifted motor brakes, etc.) will remain. This behaviour is considered as follows:

$$\begin{aligned} &P_{\mathrm{DC}}(t) + P_{\ell,\mathrm{DC}} \geq 0: \\ &\qquad P_{\mathrm{grid}}(t) = P_{\mathrm{DC}}(t) + P_{\ell,\mathrm{DC}} + P_{\ell,\mathrm{grid}} \\ &\qquad P_{\mathrm{R}}(t) = 0, \\ &P_{\mathrm{DC}}(t) + P_{\ell,\mathrm{DC}} < 0: \\ &\qquad P_{\mathrm{grid}}(t) = P_{\ell,\mathrm{grid}}, \\ &\qquad P_{\mathrm{R}}(t) = -(P_{\mathrm{DC}}(t) + P_{\ell,\mathrm{DC}}), \end{aligned}$$

where $P_{\mathrm{R}}(t)$ is the power dissipated via the brake resistor. The time integral of grid power demand over process time (first package drops (t_{start}) to completion of the tenth (t_{end})).

$$E_{\text{grid}} = \int_{t_{\text{start}}}^{t_{\text{end}}} P_{\text{grid}}(t)\ \mathrm{d}t \tag{7}$$

equals the grid energy demand of the system of a given process.
The energy consumption model is validated [13] and has a remaining model deviation of approx. 5 % over the whole spectrum of operating temperatures [18].

V. Evaluation Results

In order to evaluate our methodology, we specified and modeled the production cell example shown in Fig. 2. We evaluate the results obtained by the SCO, but not the integration with DTO, which is ongoing work.

For evaluation purposes we modeled the process described in Sect. II in SML. The model includes costs (= consumed energy) associated with the movement of the robots:

- feed robot moves from feed belt to work belt
- feed robot moves from work belt back to feed belt
- work robot moves back into idle/ready position after performing its work
- deposit robot moves from work belt to deposit belt
- deposit robot moves from deposit belt back to work belt

Each movement was divided into three phases: acceleration, movement at constant velocity, and deceleration. Additionally, we added costs for the work performed by the work robot. However, we only used a single value for this instead of subdividing it into separate steps (align with work item, track item, perform work). In total, we generated three controllers from this SML specification via synthesis. We generated (1) a random, unoptimized strategy, ignoring costs, (2) an optimized strategy using fictive values (-1 for deceleration, 1 for every other action/phase), and (3) an optimized strategy using energy values measured in a simulation driven by strategy (2). The two optimized strategies (2)+(3) are the result of applying our engineering process with two iterations. Since the DTO is still missing, the measured energy values did not change when driving the simulation with the PLC code obtained from the second iteration. Thus, no additional iterations were necessary in this experiment.

For each strategy, we generated a full PLC program and used it to drive a simulation of the production cell from Fig. 2. We measured the energy consumed or dissipated by each robot when processing 10 work items. The work items arrived at fixed intervals. We varied this interval length to simulate different work load situations. The simulation does not contain any random factors, thus we ran each experiment (combination of strategy and interval length) only once.

Table I shows the total energy E_{Grid} consumed by the robots after processing all 10 work items based on equation 7. Table II shows how much braking energy was actually dissipated as heat instead of being recuperated.

The results show that SCO can reduce the overall energy consumption, but the energy savings are relatively small. However, SCO still produced a significant effect—the main purpose of SCO is to order the robot movements in such a way that braking and acceleration phased overlap, but only integrating DTO may align the braking and acceleration phases so that the energy reuse exploited to a larger extent.

TABLE I

Total energy in kJ, consumed by all the robots

interval length	5.0 s	6.5 s	8.0 s
unoptimized strategy	26.02	25.98	25.95
optimized strategy with fictive energy values	25.47	25.21	25.17
optimized strategy including robot cell model	25.47	25.22	25.16

TABLE II

Available braking energy in kJ, which is not recuperated in our experiments

interval length	5.0 s	6.5 s	8.0 s
unoptimized strategy	9.15	9.16	9.18
optimized strategy with fictive energy values	8.62	8.41	8.40
optimized strategy including robot cell model	8.62	8.40	8.41

Table III shows what fractions of the braking energy that is dissipated as heat (see Table II) is actually lost due to poor alignment of the braking and acceleration phase. For example, the 61.5% in the bottom right cell of Table III means that in the scenario where 10 work items arrive in an interval of 8 second, and where the system is controlled by an optimized strategy, 61.5% of the braking energy is lost in an interval of the discrete control strategy where in fact an acceleration phase takes place—however, not optimally aligned. That means, due to timing issues, only a fraction of the braking energy may be reused. It is important to note that the optimized control strategies create higher amounts of braking energy that may yet be exploited by integrating DTO. Validating that DTO can in fact exploit the remaining energy saving potential is our next step, but we do not have results at the time of writing this paper.

Another interesting insight is that the total energy consumption for the optimization with fictive energy values (SCO iteration one, row 3 of Table I) and the one based on the detailed energy model of the robot cell (SCO iteration one, row 4 of Table I) is almost equivalent. This means that by choosing the fictive values (+1/-1) for acceleration and braking phases already yields a good basis for SCO. This, however, may not be always the case. In systems with different kinds of robots, implying different kinds of masses and durations of movements, a second iteration of SCO may be strictly required, or it may be required to choose fictive starting costs in a more fine-grained manner.

VI. Related Work

To the best of our knowledge, our work is the first application of synthesis techniques for production system controllers that exploit braking energy recuperation. Other controller synthesis techniques exist that target efficient motion planning for mobile robots [19], [20] or lawnmowers [21], without considering energy recuperation.

TABLE III

RELATIVE AMOUNT OF BRAKING ENERGY LOST DUE TO POOR ALIGNMENT OF CONCURRENT MOVEMENT.

interval length	**5.0 s**	**6.5 s**	**8.0 s**
unoptimized strategy	35.9 %	49.5 %	49.4 %
optimized strategy with fictive energy values	53.4 %	61.6 %	61.4 %
optimized strategy including robot cell model	53.5 %	61.6 %	61.5 %

VII. CONCLUSION

We presented a novel engineering methodology for the development of energy-efficient production system controllers. Our methodology benefits from easy to use scenario-based specifications in combination with two automated optimization techniques, SCO and DTO, in order to be able to synthesize controller software which exploits the recuperation of braking energy. These optimization techniques can be applied iteratively to achieve the best possible results.

The evaluation based on an example system with three robots shows that optimized discrete control strategy determined by applying the SCO step is more efficient, but that the energy saving is relatively small. However, a more important result is that SCO procedure indeed shifted acceleration and deceleration phases such that more braking energy is generated during the acceleration phases other robots. This leads us to believe that by integrating DTO, we will be able to more drastically improve energy savings in the future.

We are working on integrating the existing DTO techniques [13], [22] into our methodology. We also plan to study how well the technique scales with more complex systems and variations in the production processes.

REFERENCES

[1] M. Dai, D. Tang, A. Giret, M. A. Salido, and W. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 5, pp. 418–429, 2013.

[2] C. Upton, F. Quilligan, C. García-Santiago, and A. González-González, "Energy efficient production planning," in *Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services*, ser. IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 2013, vol. 397, pp. 88–95.

[3] F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *Journal of Cleaner Production*, vol. 67, no. 0, pp. 197–207, 2014.

[4] Hansen, C., Kotlarski, J., and Ortmaier, T., "Optimal motion planning for energy efficient multi-axis applications," *International Journal of Mechatronics and Automation (IJMA)*, vol. 4, no. 3, pp. 147–160, 2014.

[5] A. Mohammed, B. Schmidt, L. Wang, and L. Gao, "Minimizing energy consumption for robot arm movement," *Procedia CIRP*, vol. 25, no. Supplement C, pp. 400 – 405, 2014, 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212827114010865

[6] E. Glorieux, S. Riazi, and B. Lennartson, "Productivity/energy optimisation of trajectories and coordination for cyclic multi-robot systems," *Robotics and Computer-Integrated Manufacturing*, vol. 49, no. Supplement C, pp. 152 – 161, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584516303155

[7] D. Meike and L. Ribickis, "Recuperated energy savings potential and approaches in industrial robotics," in *2011 IEEE International Conference on Automation Science and Engineering*, Aug 2011, pp. 299–303.

[8] Hansen, C., Öltjen, J., Meike, D., and Ortmaier, T., "Enhanced approach for energy-efficient trajectory generation of industrial robots," in *Proceedings of the 2012 IEEE International Conference on Automation Science and Engineering (CASE)*, 2012, pp. 1–7.

[9] J. Greenyer, D. Gritzner, T. Gutjahr, F. König, N. Glade, A. Marron, and G. Katz, "Scenariotools – a tool suite for the scenario-based modeling and analysis of reactive systems," *Science of Computer Programming*, vol. 149, no. Supplement C, pp. 15 – 27, 2017, special Issue on MODELS'16. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642317301442

[10] K. Chatterjee, W. Dvorák, M. Henzinger, and V. Loitzenbauer, "Conditionally Optimal Algorithms for Generalized Büchi Games," in *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), P. Faliszewski, A. Muscholl, and R. Niedermeier, Eds., vol. 58. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 25:1–25:15. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/6440

[11] D. Gritzner and J. Greenyer, "Synthesis of Cost-optimized Controllers from Scenario-based GR(1) Specifications," in *Modellierung 2018*, I. Schaefer, D. Karagiannis, A. Vogelsang, D. Méndez, and C. Seidl, Eds. Bonn: Gesellschaft für Informatik e.V., 2018, pp. 167–182.

[12] ——, "Generating Correct, Compact, and Efficient PLC Code from Scenario-based Assume-Guarantee Specifications," in *Proceedings of the 4th International Conference on System-Integrated Intelligence: Challenges for Product and Production Engineering, SysInt 2018 (to appear))*, 2018.

[13] Z. Ziaukas, K. Eggers, J. Kotlarski, and T. Ortmaier, "Optimizing ptp motions of industrial robots through addition of via-points," in *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, 2017, pp. 527–538.

[14] D. Harel and R. Marelly, *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003.

[15] P. Hamon, M. Gautier, and P. Garrec, "Dynamic Identification of Robots With a Dry Friction Model Depending on Load and Velocity," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 6187–6193.

[16] C. T. Johnson and R. D. Lorenz, "Experimental identification of friction and its compensation in precise, position controlled mechanisms," *IEEE Transactions on Industry Applications*, vol. 28, no. 6, pp. 1392–1398, 1992.

[17] J. Swevers, C. Ganseman, D. B. Tukel, J. de Schutter, and H. V. Brussel, "Optimal robot excitation and identification," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 730–740, Oct 1997.

[18] K. Eggers, E. Knöchelmann, S. Tappe, and T. Ortmaier, "Modeling and experimental validation of the influence of robot temperature on its energy consumption," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT)*, 2018.

[19] Y. Wang, N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "Task and motion policy synthesis as liveness games," in *Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS'16. AAAI Press, 2016, pp. 536–540. [Online]. Available: http://dl.acm.org/citation.cfm?id=3038594.3038661

[20] R. Ehlers and V. Raman, "Slugs: Extensible GR(1) synthesis," in *Computer Aided Verification*, S. Chaudhuri and A. Farzan, Eds. Cham: Springer International Publishing, 2016, pp. 333–339.

[21] M. Randour, "Automated synthesis of reliable and efficient systems through game theory: A case study," in *Proceedings of the European Conference on Complex Systems 2012*, T. Gilbert, M. Kirkilionis, and G. Nicolis, Eds. Cham: Springer International Publishing, 2013, pp. 731–738.

[22] K. Eggers, Z. Ziaukas, S. Tappe, and T. Ortmaier, "On the relationship of travel time and energy efficiency of industrial robots, proceedings of the international conference on industrial, enterprise, and systems engineering," in *International Conference on Industrial, Enterprise, and System Engineering (ICoIESE)*, 2017.