Robotics and Autonomous Systems 62 (2014) 267–280

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Constraint optimization model of a scheduling problem for a robotic arm in automatic systems

Ewa Kolakowska*, Stephen F. Smith, Morten Kristiansen

Aalborg University, Department of Mechanical and Manufacturing Engineering, Fibigerstraede 16, 9220 Aalborg, Denmark
Carnegie Mellon University, The Robotics Institute, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

HIGHLIGHTS

- We model robotic painting as a scheduling problem using constraint optimization.
- We present software architecture for automatic generation of robot programs.
- We measure the processing time, painting quality and computation time.
- Longer processing time is a tradeoff in paying attention to quality.
- Multiple solutions can decrease processing time and improve painting quality.

ARTICLE INFO

Article history:
Received 24 September 2012
Received in revised form
20 August 2013
Accepted 13 September 2013
Available online 8 October 2013

Keywords:
Scheduling
Painting quality
Partitioned goals
Constraint programming

ABSTRACT

In this paper, we investigate the problem of scheduling a 6 DOF robotic arm to carry out a sequence of spray painting tasks. The duration of any given painting task is process dependent and fixed, but the duration of an "intertask", corresponding to the process of relocating and reorienting the robot arm from one painting task to the next one, is influenced by the order of tasks and must be minimized by the scheduler. There are multiple solutions for reaching any given painting task and tasks can be performed in either of two different directions. Further complicating the problem are characteristics of the painting process application itself. Unlike spot-welding, painting tasks require movement of the entire robot arm. In addition to minimizing intertask duration, the scheduler must strive to maximize painting quality and the problem is formulated as a multi-objective optimization problem. The scheduling model is implemented as a stand-alone module using constraint programming, and integrated with a larger automatic system. The results of a number of simulation experiments with simple parts are reported, both to characterize the functionality of the scheduler and to illustrate the operation of the entire software system for automatic generation of robot programs for painting.

1. Introduction

Robotic arms are now widely used in industry for performing different types of manufacturing processes, including welding, assembly and surface treatment. Yet, programming of industrial robots is still often done manually. This means that performance parameters such as programming time of the robot, processing time of the robot and quality of the product transformed by the robot can all change, as they depend on the individual skills of the operator. Manual programming of industrial robots is also very expensive: in addition to the worker's wage, it consumes significant time (sometimes up to 1 day) of the robot cell or, in some cases, the entire production line, and the complexity of parts together with trends toward small batch sizes only add to the programming costs. For all of these reasons it is in the interest of the industry to automate the generation of robot programs.

Automatic generation of the robot program can be decomposed into three main sub-problems: *Task Planning* (where the tasks are described in a well defined coordinate system and assigned process dependent parameters), *Task Scheduling* (where the tasks are sequenced according to some objective criteria) and *Motion Planning* (where collision-free robot specific trajectories are generated). The task scheduling model presented in this paper does not depend on the methods for task planning and motion planning and therefore these two topics are not further described. The task scheduling model is generic. It does not depend on any special characteristics of the robot and it could be used for any painting machine. It relies

* Corresponding author at: Aalborg University, Department of Mechanical and Manufacturing Engineering, Fibigerstraede 16, 9220 Aalborg, Denmark. Tel.: +45 30228660; fax: +45 9815 3030.
E-mail addresses: ewa@m-tech.aau.dk (E. Kolakowska), sfs@cs.cmu.edu (S.F. Smith), morten@m-tech.aau.dk (M. Kristiansen).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.09.005

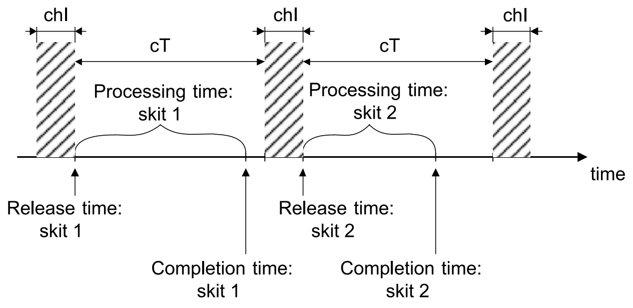

Fig. 1. An example of a time line for a robot cell.

only on the information passed in the input tasks, which is generated by the task planner and described later in this article.

The problem considered here is for a single cell with one 6 DOF spray painting robot (although the solution proposed generalizes straightforwardly to multiple cells and multiple robots per cell). The parts entering the cell are hanging on hooks, attached to a metal hollowed bar, which is also referred to as a skit. The skit is mounted on an on–off conveyor belt, implying that parts are available for painting by the robot at a specific release time and the skit does not move while in the robot cell. The conveyor is off for a predefined cycle time, *cT*, which determines the upper bound on the solution produced by the scheduler. The upper bound specifies the maximum feasible duration of the schedule. Cycle time is followed by a so-called changeover interval, *chI*, during which one skit leaves the robot cell and another one enters, see Fig. 1. The time during which a robot is working on a skit is called the processing time. Even though it does not affect the cycle time of the conveyor, its duration is important because of its influence on the total throughput. Shorter processing times give a high level shop order planner the freedom to assign more parts to a skit.

In this paper, we focus on scheduling the tasks assigned to a particular skit. The *tasks* should be understood as value-added actions, performed by the robot, such as welding or painting. They are executed by the linear or circular motion of the tool. In the painting process considered here, tasks are associated with paint strokes. A paint stroke carries information about the type of motion the paint gun should follow while it is turned on, which depends on the geometry of the surface to be painted [1]. In this article we consider only parts with planar surfaces and they are painted by the linear motions of the paint gun, which are represented by the black solid lines in Fig. 2. The paint stroke also carries information about the start and end positions and orientation of the paint gun, referred to as the goal placements and defined in the global coordinate frame. This means that the considered paint strokes are not only horizontal or vertical, but can have any orientation in the global coordinate frame. Each paint stroke has two potential start goal placements, given that it is possible to perform the paint stroke in two opposite directions, see Fig. 2. If point *A* is chosen for a start goal placement, then in this case the robot tool moves from the right to the left side. If point *C* is chosen for a start goal placement, then the robot tool moves from the left to the right side.

Since the goal placement describes both the position and orientation of the paint gun, it is important to note that:

$$position(A) = position(D), \qquad orientation(A) \neq orientation(D),$$
$$position(B) = position(C), \qquad orientation(B) \neq orientation(C),$$

and therefore:

$$goalPlacement(A) \neq goalPlacement(D),$$
$$goalPlacement(B) \neq goalPlacement(C).$$

Apart from its goal placements a paint stroke is also specified by a set of parameters, such as the distance from the nozzle to the surface that needs to be painted, the working angle, the forward angle and the speed of the paint gun. The speed and orientation of the paint gun along the paint stroke should be kept constant in order to achieve uniform thickness of the paint layer. Thus the duration of a paint stroke does not depend on the sequence of paint strokes, the painting direction or the robot they are assigned to. It cannot be used as a parameter for optimization. Instead, the sum of durations of the paint strokes assigned to a specific robot can be used by the scheduler as the lower bound on the processing time of that robot.

It is possible that not all paint strokes generated by the task planner can actually be performed by the robot, and those that cannot are deleted in the subsequent process of motion planning. A paint stroke cannot be executed if it is located outside the robot workspace or if the robot cannot reach a paint stroke because of singularities. Other reasons for deleting a paint stroke are to avoid collisions of the robot with the painting cell, the conveyor line or the parts that are hanging on the skit. The parts associated with any paint strokes that do get deleted for infeasibility reasons are missing some paint. Whether such parts are accepted or not depends on the customer's requirements and the number of deleted paint strokes. In cases where the parts are not accepted, they are either removed from production as waste or retrieved in a so called touch-up process, where the operator manually corrects for the deficiencies in the painted surface. Tasks, that are not deleted by the motion planner and thus can be performed by the robot, must be sequenced to produce a schedule.

The scheduling process is complicated by the fact that there are different ways to perform every task and the choice of solution for a given task will impact the overall skit processing time. Two alternative solutions are due to the pair of potential start goal placements for each paint stroke. Further complexity follows from the fact that a start goal placement can be reached by several joint configurations of the robot arm. A start goal placement, associated with the particular joint configuration, is referred to as a *start goal* throughout the paper. For a 6 DOF robot there can be up to 16 different solutions for reaching a certain goal placement, depending on the position of the goal placement in the robot workspace and the way the robot is built [2]. However the number of solutions for the painting application is reduced due to the fact that the orientation of the paint gun must be kept constant along the paint stroke at the same time as the robot trajectories need to be collision free. The actual number of solutions varies for different painting problems and methods used by the motion planner. Fig. 3

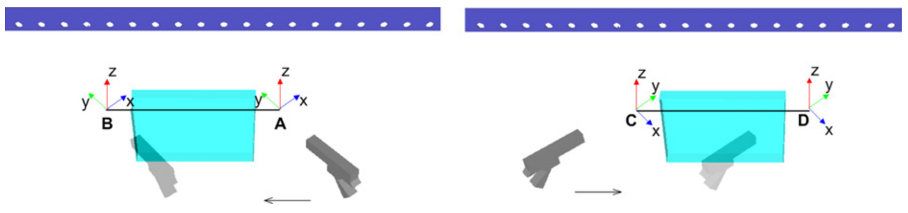

Fig. 2. An example of a paint stroke (task) with its two potential start goal placements (*A* and *C*) and two potential end goal placements (*B* and *D*).

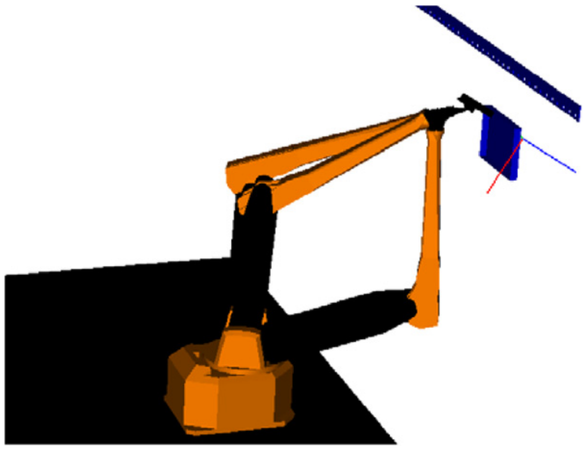

Fig. 3. Goal configurations of the robot.

shows three different joint configurations, all of which place the tool with the same position and orientation.

Both the choice of solution for every paint stroke and the sequence of paint strokes have an influence on the choice of intertask and its duration. An *intertask* is the joint motion of the robot between two tasks with a changing speed, where the tool is moving non-linearly in Cartesian space and not performing a painting task but instead is repositioning for the next one. Using terminology from the scheduling literature, an intertask can be understood as a setup time or a transition time. In the painting process the term intertask represents the time that is necessary for a robot to cover the distance from the end goal placement of one paint stroke to the start goal placement of the succeeding paint stroke. An intertask is used here as a parameter for schedule optimization, because by varying its duration it is possible to change the total processing time of the robot.

For many industrial problems, however, minimizing processing time and completing jobs quickly is not the only objective. For most manufacturers for example it is additionally important to achieve a certain level of product quality. In this respect, it is possible to analyze the properties of the existing equipment and processes for any relationship between scheduling decisions that might be taken and product quality. With respect to the painting process, scheduling of so-called overlapping paint strokes directly influences painting quality, which is measured here by the number of defects, such as sagging and the paint dust, shown in Fig. 4. Two paint strokes are overlapping when the coating material of one stroke is applied over or beyond the coating of the other one to achieve a homogeneous layer of paint, as shown in Fig. 5. This implies that two linear overlapping paint strokes are positioned on the same planar surface and they are parallel.

For paint strokes that are positioned on an inclined surface, the wet paint coat will flow down under its own weight with cohesive forces opposing this flow. The flow depends on the drying properties of paint and can be the source of defects, if the overlapping coat is applied either too fast or too late. The flow can be controlled indirectly by changing the intertask durations between overlapping paint strokes and therefore it is influenced by the chosen solutions for overlapping paint strokes and their schedule. This relationship makes it important to consider the aspect of quality in scheduling, because the solution giving the shortest processing time and best utilization of resources is worthless, if the product does not meet quality specifications.

2. Relation to previous work

The idea to automate the generation of programs for spray-painting robots first appeared in 1990s. The problem was decomposed into three main subproblems: paint planning, scheduling and motion planning. However, many challenges quickly became apparent in creating suitable models of painting processes and robot motions, whether they were an attempt to describe the physics of the system or they were based on operator's knowledge and experience. Considerable research has focused on the design of task planners. Results have been presented in articles [3–7] and incorporated in the FlexPaint project [8], summarized in [9, 10,1]. The common goal of all designed task planners was to achieve a homogeneous layer of paint deposition, though in [7] the objective was also to minimize the associated process cycle time and paint waist. The task planner introduced in [4], was tested in actual painting experiments and the robot controller was then used for motion planning the painting path. The software for motion

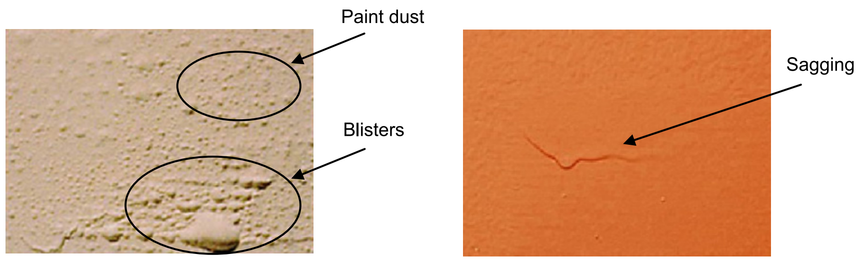

Fig. 4. Common painting defects [11,12].

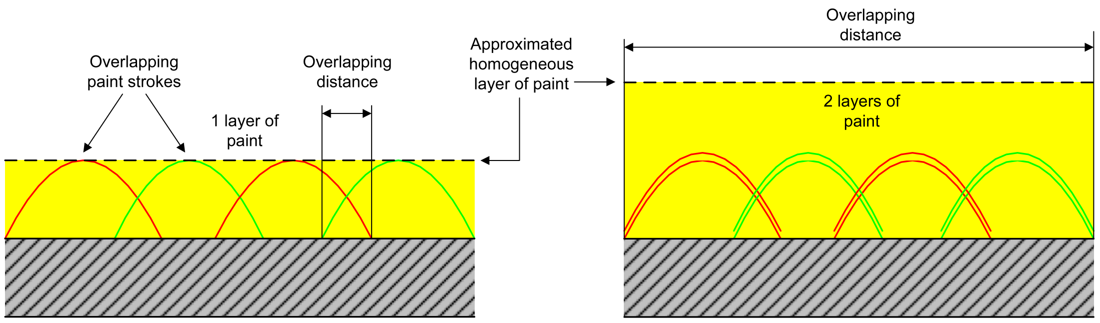

Fig. 5. Cross section of the painted surface with overlapping paint strokes.

Table 1
Summary of the reviewed articles.

Articles	Content					
	Application	Task	Solutions per task	Robot movement	Issue of quality	No tasks in exp.
[13]	Arc-welding	Line	2	Linear	✓	7, 8, 9, 10, 20, 30, 40
[14]	Robotics	Point	1	Linear		10
[15]	Spot-welding	Point	1, 5	Linear		10, 31, 50
This article	Painting	Line	2, 4, 6	Non-linear	✓	4, 6, 8, 10, 12

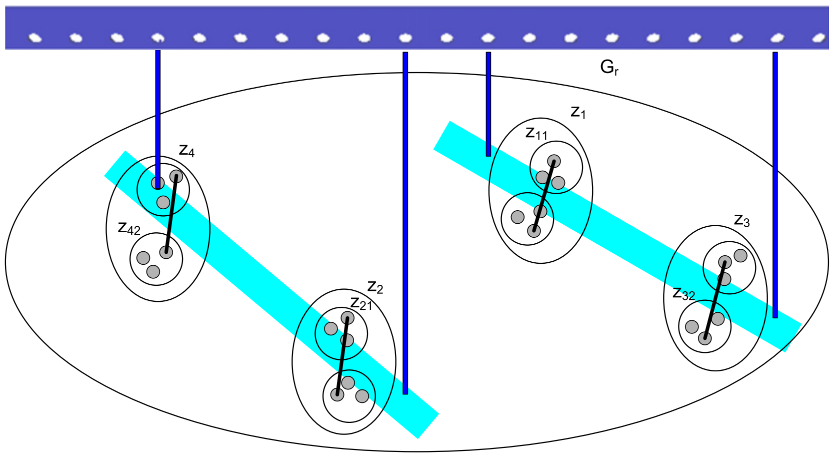

Fig. 6. Relation between physical system and the model.

planning as an integrated solution to an off-line programming tool was first described in [5] and the FlexPaint project. Though only the FlexPaint motion planner provided the method to obtain collision-free and executable robot motion for the actual robot kinematics.

No articles can be found on the development of a scheduler for spray painting robots. However schedulers have been designed for other robotic applications, such as arc-welding and spot welding [13–15]. We review this work below and relate these efforts to ours.

The scheduler described in [13] is designed for an arc-welding robot and therefore the tasks are not points, but similarly to ours they are lines, which have two potential start goal placements. However, multiple solutions due to different robot joint configurations are not considered. The objective is to minimize the time required to complete a given set of welding operations, given the assumption that along the intertasks the robot is moving in a linear mode with constant speed. The problem is modeled using an integer programming formulation and presented as a variant of a travelling salesman problem (TSP) with an additional constraint related to the sequence dependent welding quality: there has to be minimum cooling time between weld lines within a heat-affected zone. However the authors make the simplification that the heat-affected zone of a weld line is cooled uniformly and for this reason the quality constraint does not depend on the welding direction. In this work, we extend the quality model and relate it to the start time and end time of the tasks.

The scheduler described in [14] is designed to minimize the cycle time of the robot, taking into consideration the multiple solutions possible for each of the point tasks. The robot had to visit each task once, but unlike our problem, the order of visits was not critical for the overall quality performance of the product. The proposed search method was based on genetic algorithms (GA). Its performance was tested in simulation for a 3-DOF and 6-DOF robot that had to reach 10 point tasks in the three-dimensional space. A maximum number of iterations of the algorithm was set in advance, and because of this, the results obtained can only be a measure of improvement in the robot's processing time. The approach cannot guarantee the global optimum.

In [15] a solution was proposed for planning a tour of the robotic arm between point tasks. It included both the scheduler and the motion planner, integrated closely into one system. The idea was to generate a group-spanning tree (GSTree), the vertices of which represented the multiple solutions for each task. The edges, connecting any pair of tasks, were weighed by the cost, corresponding to the distance between two tasks. Because planning of collision-free paths for a 6-DOF robot was computationally heavy, the distances were initialized by their approximations, measured by the straight line segment joining two tasks and therefore not using the joint configurations of the robot arm, as we have in this work. The pro-order walk through the GSTree was performed with the objective of minimizing the overall cost of the tour, without considering the issue of product quality. The chosen edges were motion planned using the algorithm for collision avoidance and their cost was updated with the exact values. Afterward the selected path through the GSTree was checked to see if it was still optimal. In case it was not, the pre-ordered walk was performed again and the whole procedure with the chosen edges updated was repeated. The algorithm was successfully tested in simulation using 10, 31 and 50 goal groups. One experiment was conducted for non-partitioned groups (one solution for each point task) and another experiment was made for partitioned groups, which contained five different configurations for every task.

The content of the reviewed articles is summarized in Table 1, where the robot movement refers to the movement of the robot tool in Cartesian space.

3. Problem formulation

We describe the physical entities of the problem, i.e., paint strokes, different painting directions and different solutions to each paint stroke, in terms of the indices of their respective start goals. This modeling principle is explained using the example in Fig. 6, where two bars are hanging on the hooks, attached to the skit. To simplify the explanation we assume for this example that the task planner generates only four paint strokes, which in this case does not result in the complete paint coverage of the

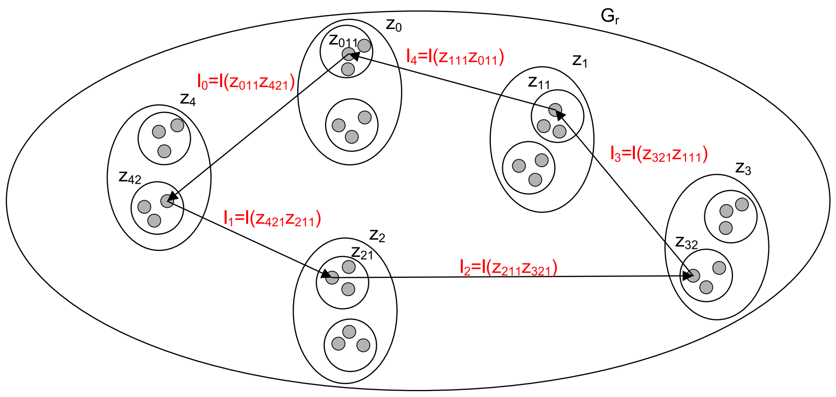

Fig. 7. An example of the start goals scheduling problem.

surfaces. Similarly to Fig. 2 the paint strokes are marked by the black solid lines and indexed by $i = 1, 2, 3, 4$. The paint stroke can be performed in two directions, $j = 1, 2$, corresponding to two potential start goal placements and symbolized in the figure by the sub-domain z_{ij} (e.g. $z_{11}, z_{21}, z_{32}, z_{42}$). The definition of direction is constrained only for overlapping paint strokes, such that overlapping paint strokes with the same value of j have the same direction in the robot coordinate frame, see Fig. 10. In this article we limit the number of distinct solutions d for reaching a given start goal placement to three (i.e., $d = 3$), which is a maximum number of solutions produced by the existing motion planner [1]. The start goals for a given paint stroke are therefore denoted by z_{ijd} (e.g. $z_{111}, z_{112}, z_{113}$). They are symbolized in the figure by the gray circles. For clearness of the figure the start goals, belonging to the same sub-domain, are not drawn on the top of each other.

Instead of using paint strokes, the scheduling problem is described here in terms of start goals. The choice of start goals for each painting activity enables the scheduler to focus strictly on how to reach the paint strokes and not on the way they are executed. The task of the scheduler is then to find the tour of the robotic arm, which traverses each paint stroke domain, z_i, once, see Fig. 7. This can be seen as an extension of a classical TSP. Due to security reasons, the tour has to start and end in the predefined home position of the robot. The home position extends the domain G_r with a dummy goal group $z_0 \in G_r$, which consists of the start goals z_{0jd}. For this particular problem the chosen start goal is z_{011}.

One objective of the scheduler is to minimize the cost of the tour. Though contrary to the TSP, the cost does not represent the length. It represents the duration of the robot movements between the start goals. Due to the non-linear tool movements along the intertasks this duration is not always proportional to the actual distance in Cartesian space between the start goals. The cost is the sum of time required to perform each paint stroke and time to cover the intertask duration to each successive start goal.

The search space of the scheduler is discrete, since the number of start goals the robot has to visit is finite. Disregarding the goal group z_0, which represents the home position, the size of search space, R_S, is described as a function of the number of tasks/paint strokes, Nt, and the number of start goals inside each goal group, $Ns = j \cdot d$. It is calculated using the following equation:

$$R_S\,(Nt, Ns) = Nt! \cdot (Ns)^{Nt}.$$

In the standard industrial setup with one spray painting robot in the robot cell the number of tasks, Nt, typically varies between 100 and 200. From the above equation it can be seen that for $Nt \in [100; 200]$ and $Ns = 6$, the size of search space, $R_S \in [6.1E + 235; 3.3E + 530]$. Fig. 8 shows how the size of search space, R_S, is effected by extending the scheduling problem with two additional paint strokes and two additional solutions; $R_S\,(X + 2, 2)$, $R_S\,(X, 4)$

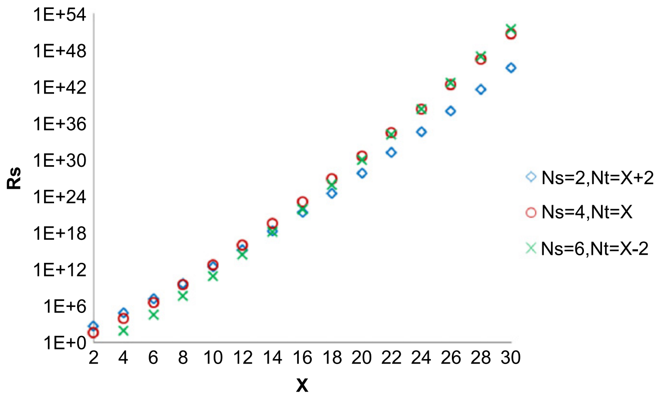

Fig. 8. Relative effect of extending the scheduling problem with two additional paint strokes and two additional solutions.

and $R_S\,(X - 2, 6)$. It is seen that for relatively small scheduling problems, $Nt \leq 6$, adding two additional paint strokes results in a bigger search space than when adding two additional solutions. It is opposite for scheduling problems where $Nt \geq 26$.

In addition to tour cost minimization, the scheduler must also cope with the painting quality issue. In this work it is required that the scheduler prevents both sags (which are due to too short of a delay between the overlapping paint strokes) and paint dust (which is due to too long of a delay between the overlapping paint strokes). The time between overlapping paint strokes is important because it imposes the drying time of paint, dT, and therefore also the flow of paint. The knowledge of this critical time window, within which sags and paint dust will not appear, can be used respectively as the *lower bound, LB,* and *upper bound, UB,* when scheduling overlapping paint strokes. The values of lower and upper bounds depend on the type of painting problem, characterized inter alia by the type of paint used, the orientation and type of surface painted, the type of paint strokes and the required thickness of the paint. The values of the lower and upper bounds have to be found experimentally prior to scheduling. This approach to modeling painting quality is based on the experience of the painting operators and some physical experiments on the painting robot, presented in [16].

When considering the issue of time separation for avoiding sags and paint dust, it is also essential to point out the role of paint stroke direction and include it in the model, which is an extension of the work presented in [13]. Performing a task involves a continuous movement of the paint gun, which has certain duration. For that reason the paint does not dry uniformly along a paint stroke. The color and physical properties of paint change in time, until the

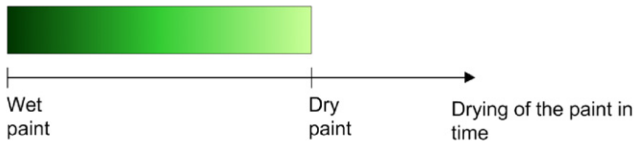

Fig. 9. A coloring scale for the drying paint.

paint is dry. Let us illustrate these changes by the coloring scale, as shown in Fig. 9.

The scale can then be used for demonstrating the relation between the choice of a paint stroke direction, the drying time of paint and appearance of defects, shown in Fig. 10. For clearness of the drawing the paint strokes are not drawn on the top of each other. Fig. 10(a) illustrates the start goal sequence, $z_{i1d} \rightarrow z_{i'2d'}$, which designates that the paint strokes are executed in opposite directions. Here the most critical area for sagging is where the drying time is shortest, encircled by the solid line. The most critical area for paint dust, encircled by the dashed line, is where the drying time is longest. In Fig. 10(b) the sequence of start goals is $z_{i1d} \rightarrow z_{i'1d'}$ and both paint strokes have the same direction. In this case, defects can occur on both ends of paint strokes.

As the paint strokes get longer and the painting speed gets slower, or as the paint strokes get shorter and the painting speed gets faster, the paint stroke directions become more essential for ensuring painting quality. Therefore they also have to be included in the scheduling model.

4. Constraint optimization model

Constraint optimization is used here as a way of modeling and solving the scheduling problem. It is attractive because it allows a strong separation to be maintained between the specification of the model and the software implementing the solver. This provides the possibility to use different solvers for the developed model and makes our solution more generic.

The main goal in constructing the model is to represent the painting problem using a structure and semantics that can easily be adapted to commercially available software programs implementing the solver for constraint optimization. The model is neither dependent on the solver's underlying search algorithms nor on the instance data. It simply specifies a given objective function and a set of constraints that must be satisfied.

In order to express two distinct constraints (see Constraint 4 below) on the same variable we extend the domain G_r with an additional dummy goal group, $z_{(Nt+1)jd} = z_{0jd}$, representing the home position.

A solution to the problem entails selection of a start goal placement for each paint task and two home positions together with an assignment of start and end times to each paint task and each home position. The binary decision variable, x_{ijd}, is introduced to express whether or not a start goal, z_{ijd}, is chosen for scheduling.

$$x_{ijd} = \begin{cases} 1 & \text{if } z_{ijd} \text{ is assigned to the robot,} \\ 0 & \text{otherwise.} \end{cases}$$

The start time and end time of task z_i are denoted by ST_i and ET_i, respectively.

The choice of start goals and the set of assigned start and end times must respect the following set of constraints.

4.1. Process constraint

• The robotic arm can traverse each paint stroke domain, z_i, only once and therefore:

$\forall i = [0, \ldots, Nt + 1]$,

$$\sum_{j=1}^{2} \sum_{d=1}^{3} x_{ijd} = 1. \tag{1}$$

4.2. Resource constraints

• A robot is a unary resource. This resource constraint dictates that the tasks of a robot do not overlap in time.

$\forall z_{ijd}, z_{i'j'd'} \in G_r$, where $x_{ijd} = 1 \wedge x_{i'j'd'} = 1 \wedge i \neq i'$,

$$(ST_i \geq ET_{i'}) \vee (ST_{i'} \geq ET_i)\,. \tag{2}$$

• The robot has limited capacity and period availability.

A robot is available to process tasks from the set G_r in the time interval [*scheduleOrigin; scheduleHorizon*]. The tightness of scheduling horizon depends on the cycle time, cT, and the changeover interval, chI, which are specified by the higher level planning and scheduling system. Therefore:

$$(ST_0 \geq scheduleOrigin) \wedge (ET_{Nt+1} \leq scheduleHorizon)\,, \tag{3}$$

where $scheduleHorizon = scheduleOrigin + cT$.

• The home position, z_{0ij}, begins the tour of the robot arm and $z_{(Nt+1)jd} = z_{0jd}$ ends the tour.

$\forall z_{ijd} \in G_r$, where $x_{ijd} = 1$,

$$(ST_0 < ST_i) \wedge (ST_{Nt+1} > ST_i)\,. \tag{4}$$

4.3. Process quality constraint

• According to the explanation in Section 3, these constraints are applied to pairs of start goals, $z_{ijd}, z_{i'j'd'} \in G_r$, that are assigned to the robot and belong to the paint strokes which are physically positioned in proximity to each other to be overlapping paint strokes. The drying time, dT, of the overlapping area should satisfy the time window requirement, $dT \in [LB; UB]$. For overlapping paint strokes $z_{ijd}, z_{i'j'd'} \in G_r$ such that $x_{ijd} = 1$ and $x_{i'j'd'} = 1$, we designate $overlapping(x_{ijd}x_{i'j'd'}) = 1$ if this is the case and assume that this information is communicated to the task scheduler by the task planner.

As the paint does not dry uniformly along the paint strokes, we formulate different sets of constraints for the two cases when

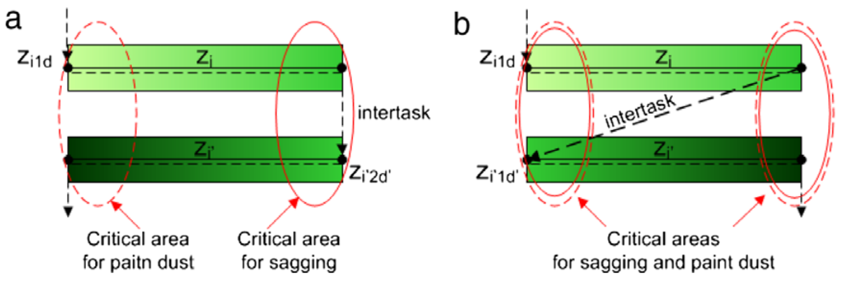
a
Zi1d
Zi
Zi'
intertask
Zi'2d'
Critical area for paitn dust
Critical area for sagging
b
Zi1d
Zi
intertask
Zi'
Zi'1d'
Critical areas for sagging and paint dust

Fig. 10. Drying time of overlapping paint strokes.

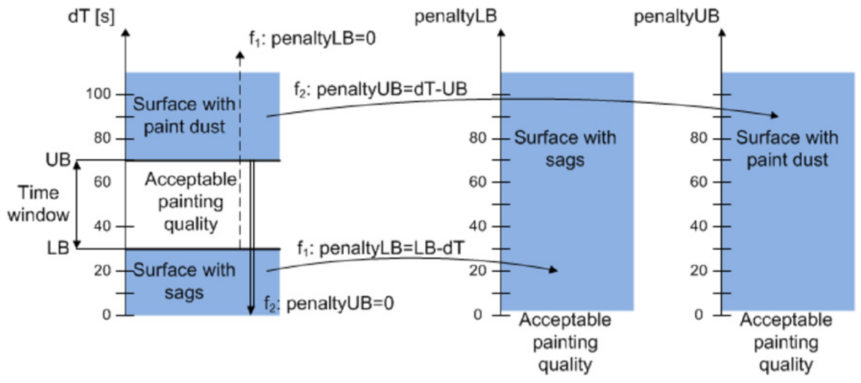

Fig. 11. Functions f_1 and f_2 mapping the drying time to the penalty.

the paint strokes have the same directions ($j = j'$) and opposite directions, $j \neq j'$.

If: $x_{ijd} = 1 \wedge x_{i'j'd'} = 1 \wedge overlapping(x_{ijd}x_{i'j'd'}) = 1 \wedge i \neq i' \wedge j \neq j'$ (corresponding to Fig. 9a), then $\forall z_{ijd}, z_{i'j'd'} \in G_r \wedge z_{ijd} \rightarrow z_{i'j'd'}$,

$$dT : ST_{i'} - ET_i \geq LB \quad \text{to prevent sagging,} \tag{a}$$
$$dT : ET_{i'} - ST_i \leq UB \quad \text{to prevent paint dust.} \tag{b}$$

If: $x_{ijd} = 1 \wedge x_{i'j'd'} = 1 \wedge overlapping(x_{ijd}x_{i'j'd'}) = 1 \wedge i \neq i' \wedge j = j'$ (corresponding to Fig. 9b), then $\forall z_{ijd}, z_{i'j'd'} \in G_r \wedge z_{ijd} \rightarrow z_{i'j'd'}$,

$$dT : ST_{i'} - ST_i \geq LB \quad \text{to prevent sagging,} \tag{c}$$
$$dT : ST_{i'} - ST_i \leq UB \quad \text{to prevent paint dust,} \tag{d}$$
$$dT : ET_{i'} - ET_i \geq LB \quad \text{to prevent sagging,} \tag{e}$$
$$dT : ET_{i'} - ET_i \leq UB \quad \text{to prevent paint dust.} \tag{f}$$

It can happen that for some problems the durations of tasks and intertasks are so long that it is impossible to satisfy either constraint (b) or constraints (d) and (f). Motivated by the particular application, where the best possible solution is better than no solution, we apply here constraint relaxation. For this purpose we consider the painting quality constraint as soft and formulate penalty variables, *penaltyLB* and *penaltyUB*, for inclusion in the objective function. These penalty variables quantify the performance of the scheduler in terms of painting quality by measuring the magnitude of the violation of the time window requirement. The larger the violation, the higher the penalty value. The specific relationship between these two entities is specified by the functions, f_1 and f_2, mapping the time delay, dT, to the *penaltyLB*, when the violation results in sagging, or to the *penaltyUB*, when the violation results in paint dust. The principle is shown in Fig. 11.

The mapping functions, f_1 and f_2, are weighted equally, because in this case there is no preference between the surface with sags and the surface with paint dust. The mapping functions are used here as the soft constraints, related to painting quality. They are formulated in the following way:

$\forall z_{ijd}, z_{i'j'd'} \in G_r \wedge z_{ijd} \rightarrow z_{i'j'd'}$,

if $(x_{ijd} = 1 \wedge x_{i'j'd'} = 1 \wedge overlapping(x_{ijd}x_{i'j'd'}) = 1 \wedge i \neq i')$

if $(j \neq j')$

$$penaltyLB_{ii'} = \max\left[LB - (ST_{i'} - ET_i), 0\right];$$
$$penaltyUB_{ii'} = \max\left[(ET_{i'} - ST_i) - UB, 0\right];$$

if $(j = j')$

$$penaltyLB_{ii'} = \max[LB - (ST_{i'} - ET_i), LB - (ET_{i'} - ET_i), 0];$$
$$penaltyUB_{ii'} = \max[(ST_{i'} - ST_i) - UB, (ET_{i'} - ET_i) - UB, 0]. \tag{5}$$

When $penaltyLB_{ii'} = 0$, the drying time, dT, between two overlapping paint strokes, i and i', is within the range marked by the dashed arrow in Fig. 11, where no sagging appears. When $penaltyUB_{ii'} = 0$, the drying time, dT, between two overlapping paint strokes, i and i', is within the range marked by the double arrow in Fig. 11, where no paint dust appears.

4.4. A multiple objective function

As indicated earlier, one objective is to minimize the tour of the robot arm through all the goal groups, $z_i \in G_r$. This is equivalent to finding the earliest end time of the last assigned start goal. Another objective is to minimize violations of the time window requirement for the overlapping paint strokes. This is achieved by assigning start goals to the robot that minimize the sum of all upper and lower bound penalties. The objective function is formulated as follows:

$\forall i, i' \in G_r \wedge i \neq i'$,

$\forall j \in \{1, 2\}$,

$\forall d \in \{1, 2, 3\}$,

$$minimize \left[w_1 \cdot \frac{c_2}{c_1} \cdot \max\left(ET_i \cdot x_{ijd}\right) + w_2 \cdot \sum_{i=0}^{Nt+1} \sum_{i'=0}^{Nt+1} (penaltyLB_{ii'} + penaltyUB_{ii'}) \right], \tag{6}$$

where:

- w_1, w_2 weight different objectives according to their importance. They are nonnegative and sum up to one [17]. The values of the weights rely on the subjective judgment of the decision-maker.
- c_1, c_2 represent the order of magnitude for objective 1 and objective 2, respectively. Their values need to be determined through the experiments. The expression $\frac{c_2}{c_1}$ is used for scaling so that both objectives are of the same order of magnitude.

In this work the objective function is simplified to the case where $c_1 = c_2 = 1$.

5. Implementation

The constraint optimization model just described is implemented as a stand-alone module, called *'Schedule tasks'*, which allows its reuse in other applications. Here it is implemented as

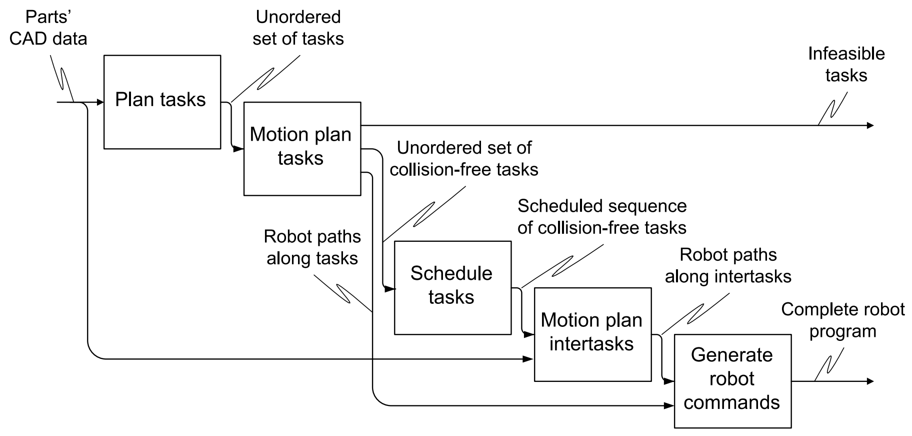

Fig. 12. Implemented architecture.

a part of a broader software system for automatic generation of robot programs for the painting process. The overall functional diagram of the designed system architecture is described based on the function modeling methodology IDEF0, as shown in Fig. 12, where the functions are executed sequentially from the left to the right.

Motion planning of the collision-free robot trajectories is relatively time consuming. Therefore the durations of all possible intertasks, considering the multiplicity of the solutions for each scheduling task, are roughly estimated in the 'Schedule tasks' function without considering changing speed of each joint, acceleration/deceleration of the joints and model of collision avoidance. The estimation approach uses the values of the rotational distance and the maximum rotational speed, $v_{\max}[i]$, for each joint i of the robot arm:

$\forall i \in [1, \dots, 6]$

$$itDuration = \max \frac{1.5 \cdot |q_2[i] - q_1[i]|}{v_{\max}[i]},$$

where the vectors $(q_1[1], q_1[2], \dots, q_1[6])$ and $(q_2[1], q_2[2], \dots, q_2[6])$ are the configurations of the end goal placement of one task and the start goal placement of another task, respectively. The value 1.5 is determined by juxtaposing two equations: the equation for the maximum rotational speed and the polynomial function describing the change of rotational speed [16]. This estimation approach does not model the intertask durations as well as the motion planner does and therefore there is a chance that the data used by the scheduler is less precise. Motion planning of the collision-free robot trajectories is performed only for the intertasks chosen by the scheduler.

The overall performance of the painting system depends on the implementation of the whole system for automatic generation of robot programs for painting process, shown in Fig. 12, including the number of paint strokes deleted by the motion planner and the estimation approach for the durations of intertasks, which are part of input data to the scheduling model. However in this article we refer to the issue of optimality only with respect to the performance of the scheduler and not the entire system.

Within the chosen architecture, task planning is performed using the *Inropa Basic 3* commercial program. Its control strategies enable painting parts with complex surfaces, such as cumbers and cavities. The tasks and intertasks are planned for collision-free robot movements by the *Amrose Robotics* software [1], which also provides functionality for generating robot commands. The graphical interface of these two software programs is shown in Figs. 14 and 15. The task scheduler is implemented in *ILog OPL Studio v.3.6*. By default *ILog OPL* exploits a depth-first search to explore the search tree and this is the algorithm applied in this paper. Since this is a complete search strategy, the returned solution is guaranteed to be the optimal solution, as defined by our objective function.

It was discovered, that ILog OPL can only handle travel minimization problems where the input data satisfies the triangular inequality. This is not the case in robotic painting. The robotic motions related to intertask repositioning are performed by point-to-point movements. In the point-to-point mode the paint gun does not have a fixed speed. Therefore covering a short distance can possibly take longer time than covering a longer distance, which violates triangular inequality requirement. To overcome this obstacle the input data to the scheduler was preprocessed, using the Floyd–Warshall algorithm [18] to find the shortest path between any pair of start goals.

6. Design of simulation experiments

In this section, we report the experiments we have performed using the implementation of our constraint optimization model within the system shown in Fig. 12. All results given below have been obtained on a 2.66 GHz Intel Xeon computer with 6 GB of memory running Windows 7 Professional. All times are in seconds, with a resolution 0.1.

The overall system to be analyzed is summarized in Fig. 13 and described based on the IDEF0 terminology.

The three input variables are: the number of goal groups (paint strokes and 2 home positions), $Ng = Nt + 2$, the number of start goals inside each goal group (number of distinct solutions for each paint stroke), $Ns = j \cdot d$, and the painting problem. The painting problem is related to the parallel paint strokes and planar surfaces, which are vertical and hence constitute the worst case scenario concerning painting quality. The mechanism signal, Nc, is representing the type of the constraint optimization model used. The notation should be understood in the following way: $Nc = 3$, describes the model implementing 3 constraints, marked as (1)–(3) in the previous section. In case the scheduler is running the simplified models described by $Nc = 2$, $Nc = 3$, $Nc = 4$, the following objective function is used:

$$minimize \lfloor \max(ET_i \cdot x_{ijd}) \rfloor. \quad (7)$$

In the case of the full model, $Nc = 5$, the objective function described previously (Eq. (6)) is assumed. The scheduler's setting signals: weights, $w1$ and $w2$, and the time window requirement, $[LB; UB]$, are both related to the process quality constraint and therefore they are only relevant for the constraint optimization model: $Nc = 5$. The other setting signal is the tightness of the

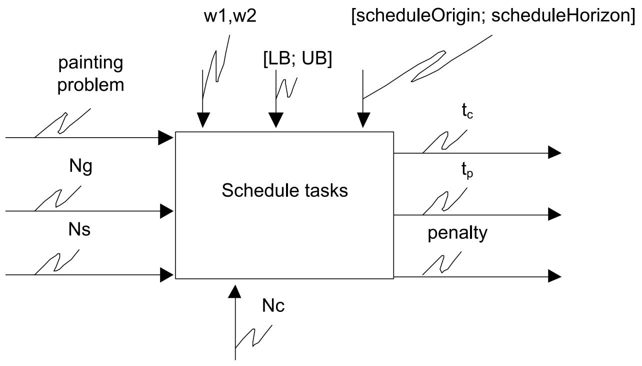

Fig. 13. System simulated in the experiments.

schedule, *[scheduleOrigin; scheduleHorizon]*.
For $Nc < 3$ scheduleHorizon $= \infty$.

We measure the performance of the scheduler along three dimensions: the computation time of the solver, t_c, the estimated processing time of the robot (or equivalently the makespan of the robot's schedule), t_p, and the total value of the penalty, *penalty*, which is indicative of the number and extent of painting defects. The performance variables are the outputs from the scheduler and can be described by the following equations:

$$t_c = f\,(Ng, Ns, Nc)\,,$$
$$t_p = f\,(Ng, Ns, Nc)\,,$$
$$penalty = f\,(Ng, Ns, Nc)\,.$$

Though for $Nc = 5$, *tp* and *penalty* are also functions of the weights, $w1$ and $w2$. For reading simplicity the above equations do not include the fact that the output variables are also functions of the painting problem.

We perform two series of simulation experiments. For both of them we assume that for all overlapping paint strokes the time window requirement has the same values, [20; 43], which are determined for the given painting problem experimentally [16] and prior to scheduling. We use the tightness of the scheduling horizon, [0; 1750], which is derived from the cycle time, $cT \approx$ 30 min. The goal of the first series of experiments is to understand the relationships between input variables, mechanism signal and output variables. The input variables, *Ng* and *Ns*, mechanism signal, *Nc*, and their levels are shown in Table 2.

We perform simulation experiments according to the plan for full factorial design over the independent variables, *Ng*, *Ns*, *Nc*, with mixed level fractions [19]. We conduct one run for each combination of levels. For $Nc = 5$ we assume that the weights are constant: $w1 = 0.9$ and $w2 = 0.1$.

Table 2
Levels of input variables.

Variables	Level				
	Level 1	Level 2	Level 3	Level 4	Level 5
Ng	4	6	8	10	12
Ns	2	4	6	–	–
Nc	2	3	4	5	–

The scheduling problems are generated by the system presented in Fig. 12. The input to this system is the CAD data associated with the parts to be painted, which is based on real industrial cases of painting sheets of different sizes. The sheets are chosen, because they have planar surfaces and more complex geometries rapidly generate $Ng > 12$. By choosing sheets the process quality constraint, related to overlapping paint strokes, can be evaluated. In Fig. 14 there is an example of the experimental setup with seven identical sheets, which is to test the performance of the scheduler for $Ng = 12$ (representing 10 paint strokes and 2 home positions). The size of the sheet used in this experiment requires two layers of paint, each consisting of two paint strokes overlapping 50%, giving in total 4 paint strokes. In order to generate the problem with $Ng = 12$ the sheets are positioned in the same plane and some paint strokes can stretch through few workpieces. The graphical interface of the *Inropa Basic* 3, shown in Fig. 14, is limited and enables only visualization of paint strokes, without highlighting overlapping ones. In Fig. 14 the numbers next to the paint strokes are indicating their order in sequence returned by the scheduler. The graphical interface of the *Amrose Robotics* software enables simulation of the movement of the robot and an indication when the paint gum is turned on, but not the deposition of paint on the surface and painting quality. Fig. 15 shows the screen shots from the simulation with the robot arm configuration for the selected start goals in the sequence of execution from Fig. 14. The full simulation can be accessed at Video S1 (available online at http://dx.doi.org/10.1016/j.robot.2013.09.005).

The goal of the second series of experiments is to understand the relationships between the internal settings of the scheduler, $w1$ and $w2$, and the output variables. Therefore the experiments are performed for $Nc = 5$. The values of the weights are varied as shown in Table 4. We conduct one run for each combination of values of weights. The tested case is for $Ng = 6$ and $Ns = 2$.

7. Results

The results of the first series of experiments are shown in Table 3. Maximum running time of the scheduler, $t_{c,}$ was set to 40 h. If the scheduler did not terminate within this time, the

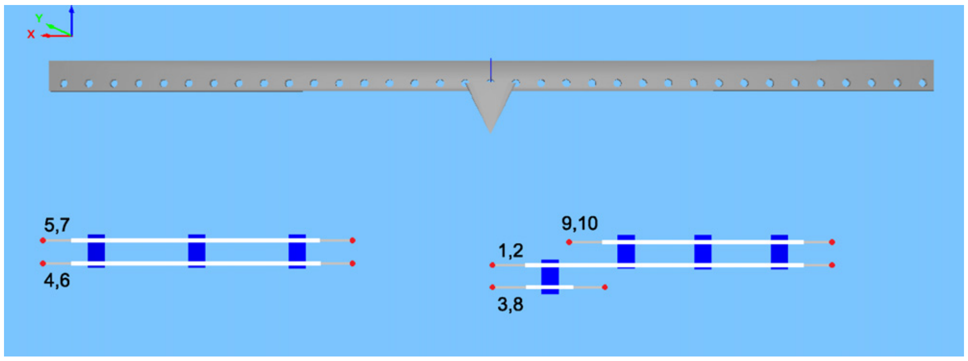

Fig. 14. Example of experimental setup for $Ng = 12$: output from *Task Planner* simulated in *Inropa Basic 3* and the paint strokes' numbers in sequence generated by the scheduler.

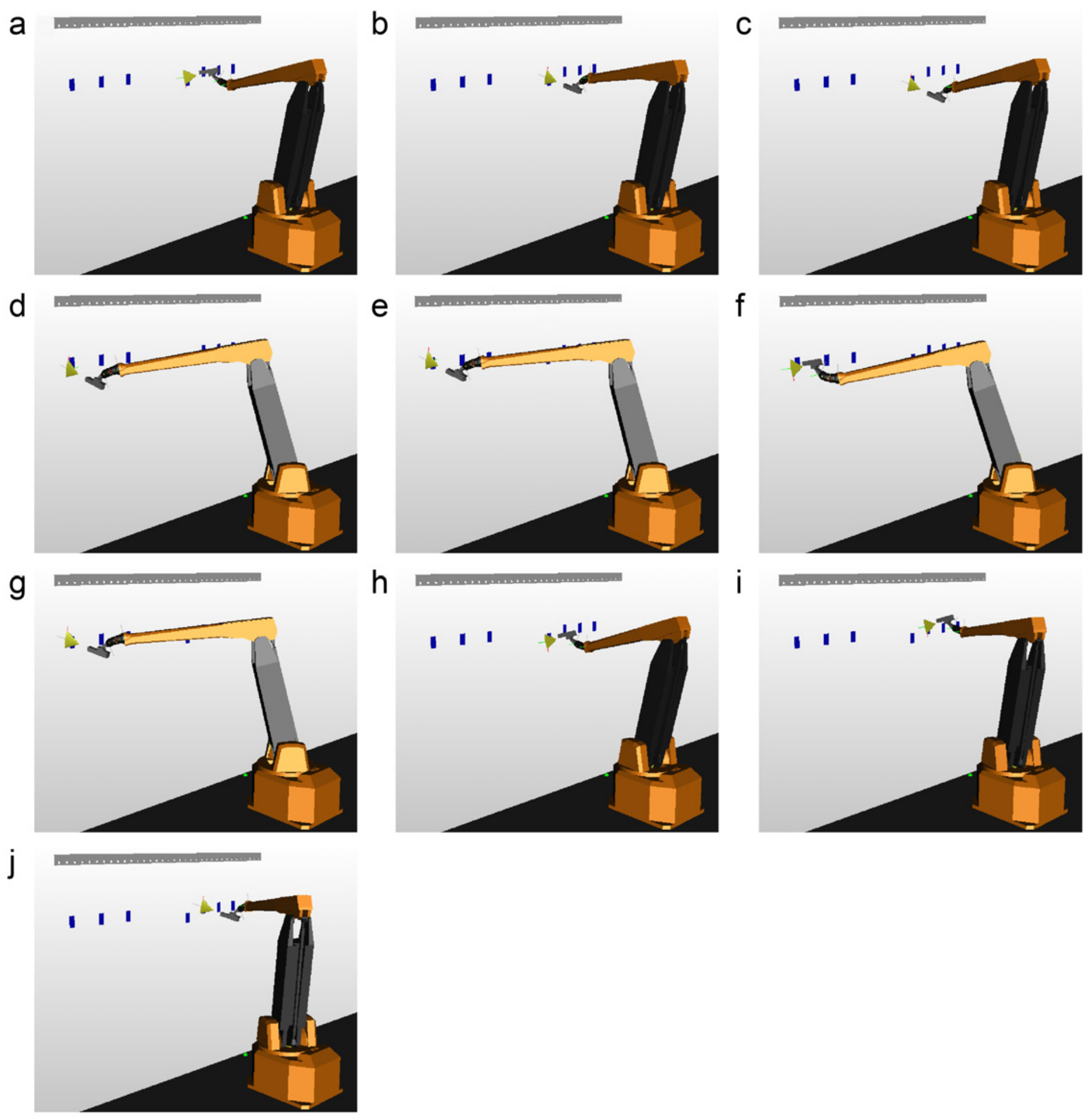

Fig. 15. Example of experimental setup for $Ng = 12$: the robot arm configuration for the selected start goals in the sequence of execution simulated in the *Amrose Robotics* software.

Table 3
Results of experiments, where pe: penalty.

Ng	*Ns*	*Nc* = 2			*Nc* = 3			*Nc* = 4			*Nc* = 5		
		Tc	Tp	pe	tc	tp	pe	tc	Tp	Pe	Tc	tp	pe
4	2	0.01	63	16	0.01	63	16	0.02	63	16	0.02	63	16
4	4	0.69	60	19	0.04	60	19	0.36	60	19	0.39	60	19
4	6	2.15	59	20	0.08	59	20	–	–	–	–	–	–
6	2	0.29	73	85	0.02	74	79	0.38	74	79	0.88	74	71
6	4	1302.67	69	83	1.03	69	83	–	–	–	–	–	–
6	6	–	–	–	18.42	68	82	–	–	–	–	–	–
8	2	83.23	182	329	5.82	182	329	170.16	191	234	519.37	196	103
8	4	15334.49	180	321	826.32	180	321	–	–	–	–	–	–
8	6	–	–	–	13945.2	180	321	–	–	–	–	–	–
10	2	219.48	93	149	4.19	94	147	139.74	94	130	1069.56	99	68
10	4	–	–	–	3707.16	90	172	–	–	–	–	–	–
10	6	–	–	–	–	–	–	–	–	–	–	–	–
12	2	–	–	–	2545.81	178	140	143934.53	185	103	–	–	–
12	4	–	–	–	–	–	–	–	–	–	–	–	–
12	6	–	–	–	–	–	–	–	–	–	–	–	–

experiment was stopped and marked as '–' in Table 3. The selected results are presented graphically in Figs. 16–18.

The results of the second series of experiments are shown in Table 4 and in Fig. 19.

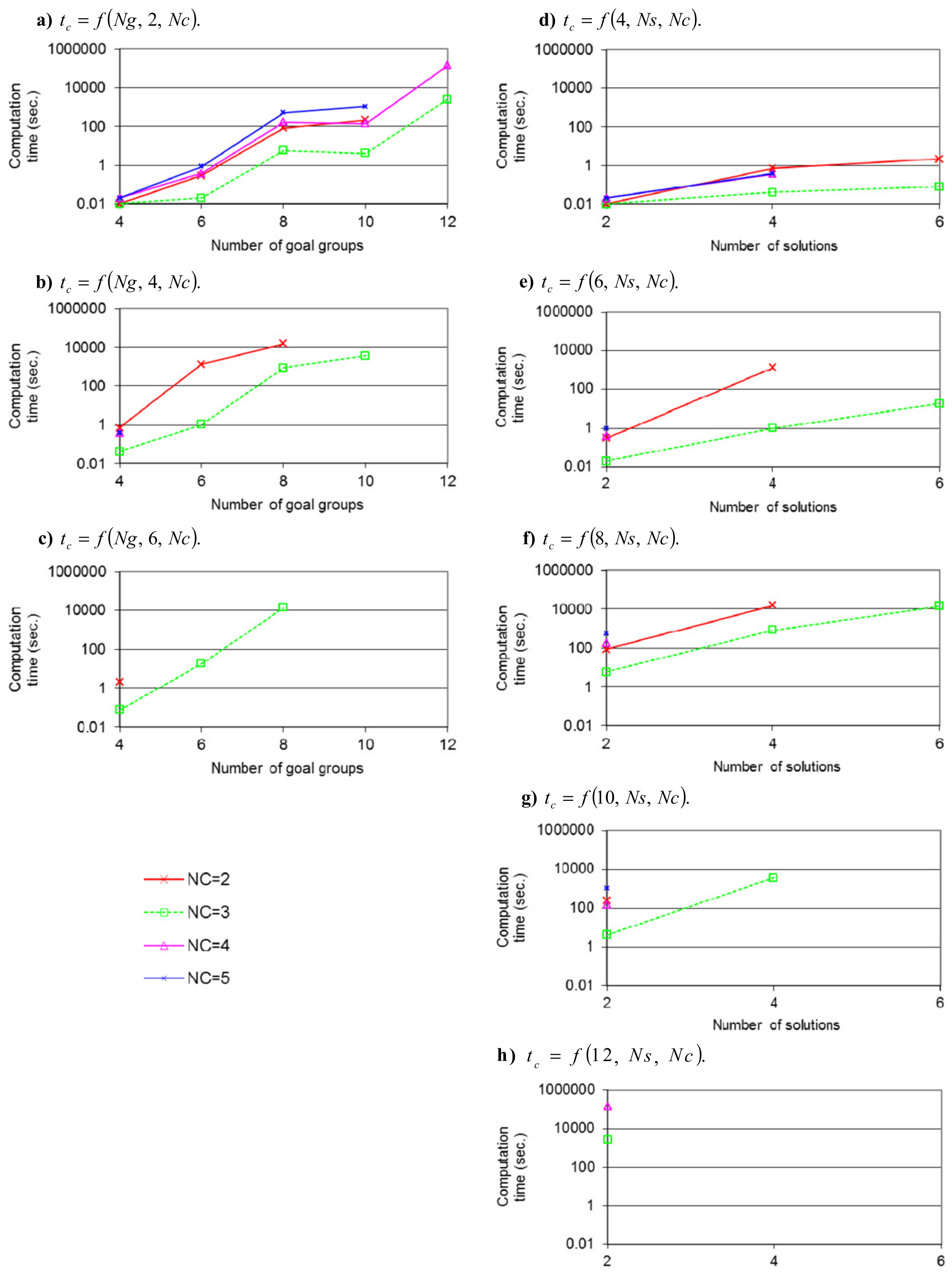

Fig. 16. Results of the experiments concerning computation time of the solver, t_c.

8. Discussion and analysis of the results

Description of the results with regards to the computation time of the scheduler, t_c*:*

Fig. 16(a): For a full model, $Nc = 5$, the computation time of the scheduler, t_c, is not affected by the specific values of the lower and upper bound parameters, *LB* and *UB*, since they are specified prior to scheduling. The computation time is rather affected by the fact of introducing an additional constraint to the scheduler and solving the multiple objective function. For a full model, $NC = 5$ with minimum number of solutions, $Ns = 2$, the scheduler can solve the problem for number of goal groups $Ng \leq 10$ (up to 8 paint strokes), within relatively short computation time, $tc \leq 100$ [s]. Extending the problem to include more paint strokes expands the search space significantly, Fig. 8. The performance of the scheduler

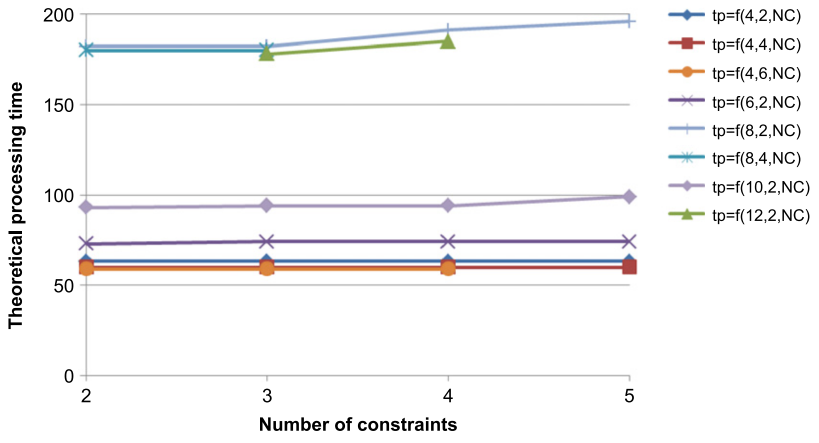

Fig. 17. Results of the experiments concerning the approximated processing time of the robot, t_p.

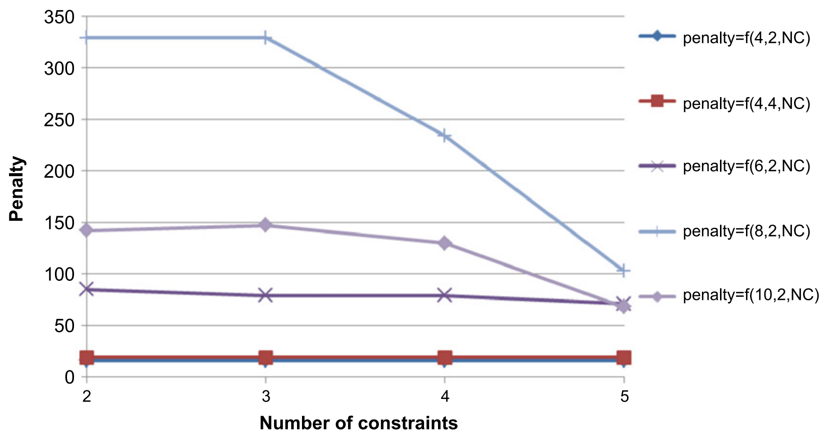

Fig. 18. Results of the experiments concerning the total value of penalty, *penalty*.

Table 4
Results of experiments with different values of weights.

	Exp.1	Exp.2	Exp.3	Exp.4	Exp.5
$w1$	0.1	0.3	0.5	0.7	0.9
$w2$	0.9	0.7	0.5	0.3	0.1
t_c	408.02	34.42	13.58	7.06	0.88
t_p	96	96	92	81	74
pe	17	17	21	37	71

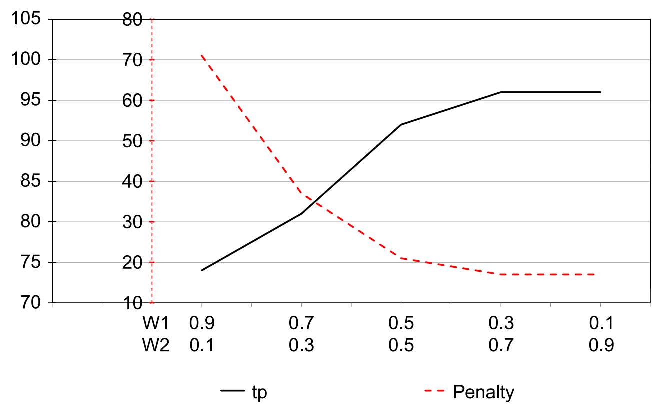

Fig. 19. Results of the experiments on effects of varied values of weights on the approximated processing time of the robot, t_p, and on the total value of penalty, *penalty*.

needs to be improved, in order to solve the industrial problems, which are at least of the size $Ng \leq 50$, within $t_c \leq 360$ s.

The results of experiments with different sized scheduling problems, show that the constraint related to the scheduling horizon, $Nc = 3$, decreases the computation time of the scheduler compared to $Nc = 2$, because it prunes many alternative solutions and this way speeds up the search.

$$t_c = f\,(Ng, Ns, 2) > t_c = f\,(Ng, Ns, 3)\,.$$

Fig. 16(a): Constraint concerning scheduling of the home position, $Nc = 4$, increases the computation time of the solver compared to $NC = 3$:

$$t_c = f\,(Ng, 2, 3) < t_c = f\,(Ng, 2, 4)\,.$$

Given the same painting problem the model $Nc = 5$ results in the longest computation time of the scheduler, t_c, because it is using the multiple objective function with two penalty variables.

Description of results with regards to the processing time of the robot, t_p:

The constraint concerning limited capacity and period availability of the scheduler does not have influence on the processing time of the robot, t_p:

$$t_p = f\,(Ng, Ns, 2) = f\,(Ng, Ns, 3)\,.$$

For some painting problems the constraint concerning scheduling of the home position can increase processing time of the robot, t_p, e.g.:

$$t_p = f\,(8, 2, 3) < t_p = f\,(8, 2, 4)\,,$$

$$t_p = f\,(10, 2, 3) < t_p = f\,(10, 2, 4)\,.$$

Both models, $Nc = 2$ and $Nc = 3$, are constraint optimization models and it is expected that they generate the same solution. In the model $Nc = 4$ it is forced that the home position is scheduled at the beginning and at the end of the sequence. In case when for $Nc = 2$ and $Nc = 3$ home positions are scheduled at the most opportune point in the sequence, the solution for $Nc = 4$ is different and possible requires a longer processing time, t_p.

For painting problems where $Ng > 4$ the process quality constraint increases the processing time of the robot, t_p:

$$t_p = f\,(Ng, Ns, 4) < t_p = f\,(Ng, Ns, 5)\,.$$

This is because for $Nc = 5$ with the multiple objective function the scheduler is not minimizing the processing time, t_p, but instead it is minimizing a tradeoff between the processing time and the penalty.

Fig. 17: Increasing the number of solutions, *Ns*, does not have a significant influence on the reduction of the processing time of the robot, t_p, e.g.:

$$t_p = f\,(4, 2, Nc) \approx f\,(4, 4, Nc) \approx f\,(4, 6, Nc)\,,$$
$$t_p = f\,(8, 2, Nc) \approx f\,(8, 4, Nc)\,.$$

This is an indication that for the considered scheduling problems the cost of traveling between the start goals belonging to different paint strokes has a relatively bigger influence on the result of objective functions (Eq. (6)) and (Eq. (7)), compared to the choice of the joint configurations for each start goal and for this reason does not influence the processing time significantly. In order to validate this hypothesis a new series of experiments should be performed, in which the linear distances between the start goals are relatively small.

Description of results with regards to painting quality, penalty: Fig. 18, $Nc = 5$: For the considered scheduling problems, modeling painting quality and including it in the objective function as described in Eq. (6) always results with less painting defects (regardless of the values tested for $w1$ and $w2$):

$$penalty = f\,(Ng, Ns, 5) < penalty = (Ng, Ns, Nc)\,,$$
$$\text{where } Nc = 1, 2, 3, 4.$$

Description of results with regards to different values of weights, $w1$ *and* $w2$: Fig. 19: By changing the values of weights in the multiple objective function, Eq. (6), we change the importance of the processing time, t_p, with respect to the painting quality, *penalty*. Longer processing time is a tradeoff when paying attention to quality. Though the slopes of curves depend on the painting problem.

It is not possible to recommend the values of the weights, $w1$ and $w2$, because they depend not only on user preferences, but also on the painting problem. The simulation experiments with different values of weights need to be performed every time the painting problem is changed.

In case the companies are interested in a specified painting quality, e.g. 99% of painting must be within specifications, the solution could be to modify the painting quality constraints (a)–(f) and use them as hard constraints in the scheduling model, instead of *penalty* in the multiple objective function. One should then define if the deviation from the specifications should be considered for every paint stroke or as a total sum of deviations from all the paint strokes.

9. Conclusions

In this paper, we modeled a new type of scheduling problem, which appears in robotic painting and welding applications with automatic generation of robot programs. In this problem the tasks have multiple solutions, the robot's intertask movements are not linear and they are performed with varying speed. It was analyzed that scheduling of overlapping paint strokes can have an influence on painting quality, represented by painting defects such as paint dust and sagging, and measured by the value of penalty. Painting quality was therefore incorporated into the objective function of the scheduler, along with the processing time of the robot. Extending the work presented in [13] we also modeled the complication that the painting quality of overlapping strokes is influenced by the fact that the paint does not dry uniformly along the paint stroke. This property was represented by two different sets of constraints: one applying to the situation when two overlapping paint strokes have the same direction and the other one applying to the situation when the overlapping paint strokes have the opposite direction. The overall scheduling problem was modeled using constraint optimization.

We evaluated our constraint optimization scheduler by conducting simulation experiments.

The goal of the first series of experiments was to get a better understanding of the relationships between the input and output variables of the constraint optimization model, as seen in Fig. 13. One of the interesting conclusions is that adding consideration of an additional solution to each paint stroke neither significantly decreases the processing time of the robot, t_p, nor does it improve the painting quality by decreasing *penalty*. This conclusion cannot be generalized though, as it depends on the painting problem, physical properties of the robot and the geometry of painting cell. Use of the full constraint model that is formulated is shown to be an effective tool in incorporating the aspect of painting quality to the scheduling problem and improving painting quality. The drawback of using this model is that the multiple objective function together with two penalty variables, *penaltyLB* and *penaltyUB*, increases the computation time of the scheduler, t_c. The results from the second series of experiments show that a longer processing time of the robot, t_p, is the tradeoff in paying attention to painting quality and that the weights in the multiple objective function can be used to balance the relative importance of processing time, t_p, and painting quality, *penalty*.

The presented simulation results were performed on the problem of painting planar, vertical surfaces, however the designed constraint optimization model can also be used on more complex surfaces, such as cavities and ribs, which still require the linear paint strokes with the methods used by the existing task planner [1]. It is also relatively easy to extend the constraint optimization model to cope with different orientations of the painted surfaces, which impose different flows of the paint. Different values of the time window requirement, $dT \in [LB; UB]$, for use by the proposed process quality constraint will have to be determined experimentally, however depending on the orientation of paint strokes. To address cases where the paint does not dry uniformly along the linear or non-linear paint strokes, a more sophisticated model of the painting process would be required and the process quality constraint should be reformulated.

There are several directions in which this work could be extended in the future. One focus could be on extending the size of the search space that the scheduler can efficiently accommodate, which is a necessity if the proposed solution should find the industrial application. Although the focus of this work has been to investigate the use of a black-box constraint solver, our experiments utilized only the basic, depth-first search procedure that is provided as ILog OPL's default solver. This procedure ensures that an optimal solution to the formulated constraint model will (eventually) be found if given enough computation time, but as our experiments have shown, it is very inefficient in this solving context. The incorporation of more sophisticated scheduling heuristics and search algorithms has enabled effective scaling in many other constraint-based scheduling applications, and application of similar techniques to the paint problem should be explored. It may also be possible to achieve scalability by imposing further structure on the overall search space, for example by partitioning tasks into sub-groups of overlapping paint strokes that must be scheduled consecutively.

A second future focus could be to investigate extension of the scheduling problem covered in this article to a multirobot environment, where the same task can be assigned to more than one resource. This requires a few extensions in the system for automatic generation of robot program, Fig. 12. Motion planning should be done for all the tasks on all the robots, in order to determine which resources are capable to perform the tasks. This will certainly imply some situations, where one task can be performed by more than one robot and therefore task assignment should be implemented after motion planning. In case that the assignment of overlapping tasks between different resources is allowed, some modifications to the process quality constraint in the scheduling model will also be necessary. Moreover the applications for multiple robots per cell with overlapping workspaces require that the motion planner is generating collision-free robot trajectories.

Finally, it would be interesting to investigate how good the rough estimates of the intertask durations that the scheduler is using for optimization are, and what impact these approximations have on the efficiency of the overall robot program generation process.

References

[1] G. Biegelbauer, A. Pichler, M. Vincze, C.L. Nielsen, K. Häusler, The inverse approch of flexpaint, in: Robotics and Automation Magazine, Vol. 12, IEEE, 2002, pp. 24–34.

[2] J.J. Craig, Introduction to Robotics, Mechanics and Control, second ed., Addison-Wesley Publishing Company, 1989.

[3] S.-H. Suh, I.-K. Woo, S.-K. Noh, Development of an automatic trajectory planning system (atps) for spray painting robots, in: Proc. of the 1991 IEEE International Conference on Robotics and Automation, 1991, pp. 1948–1955.

[4] N. Asakawa, Y. Takeuchi, Teaching spray-painting of sculptures by an industrial robot, in: Proc. of IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, 1997.

[5] E. Freund, D. Rokossa, J. Roßmann, Process-oriented approach to an efficient off-line programming of industrial robots, in: Proc. of the 24th Annual Conference of the IEEE, vol. 1, 1998, pp. 208–213.

[6] H. Chen, N. Xi, W. Sheng, M. Song, Y. Chen, Cad-based automated robot trajectory planning for spray painting of free-form surfaces, Industrial Robot 29 (5) (2002) 426–433.

[7] N.P. Atkar, A. Greenfield, C.D. Conner, H. Choset, A.A. Rizzi, Uniform coverage of automotive surface patches, The International Journal of Robotics Research (2005) 883–898.

[8] Flexpaint, 2000–2002, Efficient low volume high variant robotized painting, www.flexpaint.org.

[9] A. Pichler, M. Vincze, H. Andersen, O. Madsen, K. Häusler, A method for automatic spray painting of unknown parts, in: Proc. of the 2002 IEEE International Conference on Robotics and Automation, pp. 444–449, 2002.

[10] M. Vincze, A. Pichler, G. Biegelbauer, K. Häusler, H. Andersen, O. Madsen, M. Kristiansen, Automatic robotic spray painting of low volume high variant parts, in: Proc. of 33th International Symposium on Robotics, 2002.

[11] Standard Paint & Flooring, viewed 12 October 2013, http://www.standardpaintandflooring.com/136_.cfm.

[12] About.com, viewed 12 October 2013, http://homerepair.about.com/od/exteriorhomerepair/ss/paint_failures_5.htm.

[13] H.-J. Kim, Y.-D. Kim, D.-H. Lee, Scheduling for an arc-welding robot considering heat-caused distortion, Journal of Operational Research Society 56 (2005) 39–50.

[14] P. Zacharia, N. Aspragathos, Optimal robot task scheduling based on genetic algorithms, Robotics and Computer-Integrated Manufacturing 21 (2005) 67–79.

[15] M. Saha, T. Roughgarden, Latombe Jean-Claude, G. Sánchez-Ante, Planning tours of robotic arms among partitioned goals, The International Journal of Robotics Research (2006) 207–223.

[16] E. Kolakowska, Integrated planning and scheduling in fully automatic robot systems, Ph.D. Thesis, ISBN: 87-91200-63-6, 2012.

[17] J.D. Camm, J.R. Evans, Management Science, International Thomson Publishing, 1996.

[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, 1990.

[19] D.C. Montgomery, Design and Analysis of Experiments, fifth ed., John Wiley & Sons, 2001.

Ewa Kolakowska is an assistant professor in the Department of Mechanical and Manufacturing Engineering at Aalborg University, Denmark. In 2005 she received her MSc in the Department of Electrical Engineering at Warsaw University of Technology, Poland, and MSc in the Department of Electronic Systems at Aalborg University, Denmark. In 2007 she was a visiting graduate student at The Robotics Institute at Carnegie Mellon University, USA. She received her Ph.D. degree from the Department of Production at Aalborg University in 2009. Her research interests include scheduling of the industrial robots and design and control of the autonomous mobile field robots.

Stephen F. Smith is a Research Professor in the Robotics Institute at Carnegie Mellon University, where he heads the Intelligent Coordination and Logistics Laboratory. Dr. Smith's research focuses broadly on the theory and practice of next-generation technologies for planning, scheduling, coordination and optimization. He pioneered the development and use of constraint-based search and optimization models for solving planning and scheduling problems, and continues to investigate extensions that enable their effective use in complex, uncertain and distributed domains. He has published over 240 technical articles in these areas, and is a Fellow of the Association for the Advancement of Artificial Intelligence.

Morten Kristiansen is associate professor at the Department of Mechanical and Manufacturing Engineering, Aalborg University, Denmark, where he is teaching and researching in the area of manufacturing technology, robotics and process automation. Currently his research is in automation and modeling of the laser cutting and welding process. Previously Morten Kristiansen has worked at Aalborg University and received in 2007 his Ph.D., in the field of modeling of the welding process using different knowledge sources and artificial intelligence.