# Thesis Abstract: Path-planning for Industrial Robots Among Multiple Under-specified Tasks

**Sergey Alatartsev, Frank Ortmeier**
Computer Systems in Engineering, Otto-von-Guericke University of Magdeburg, Germany
{sergey.alatartsev, frank.ortmeier }@ovgu.de

## Introduction

Nowadays industrialized countries with their high labor costs have to rely on production automation to keep their competitive advantage. One of the most flexible and powerful automation technology available today is industrial robotics. Equipped with the right tool, standardized industrial robots can perform numerous production tasks. Since acquisition and programming of an industrial robot are very expensive, the feasibility of using robots in production facilities depends on the efficiency with which the robot can perform its task: the more production steps a robot can perform in a given time interval, the higher the production rates, as a consequence the faster the robot can compensate for its initial acquisition and programming costs, and the higher is the competitive advantage it provides to the company.

Virtually all robotics scenarios consist of two different types of robotic movements. The first category includes movements that are specifically required for the job. For example welding a seam, deburring some sharp edge or cutting a shape. These movements are typically the movements, during which the tools (e.g., a welding torch) are switched on. We call this category *effective movements* or *effective tasks*. In between two effective movements are supporting movements. Supporting movements are not directly needed for a given job. However, they are necessary to sequence one effective movement after the other. In the example of seam welding, these movements would be to move the robot from welding seam to welding seam. We call these movements *supporting movements* or *supporting tasks*. Fig. 1 illustrates this concept on a simple welding example. The two welding seams – (2) and (4) – are effective movements, while (1), (3) and (5) are supporting movements, which are only necessary to execute the effective movements.

The major characteristics that affect the efficiency of a given robot are how fast the robot can perform its tasks (effective movements), and how long it takes the robot to move into position to perform a task after having completed the previous one (support movements). While effective movements are usually understood as rigid paths (defined by the task/application), supporting movements are open to computational optimization.

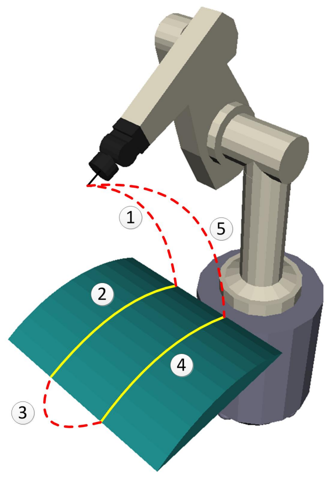

Figure 1: Example of the alternating stages

The current industrial practice is to let an engineer program both effective and supporting movements. In contrast, state-of-the-art research already provides solutions for automatically computing collision-free, supporting movements and/or optimizing the schedule of effective tasks.

In currently going PhD project, we intent to go one fundamental step further: we observe that not only supporting movements but also many effective movements are under-specified. For example, many robot tasks (e.g., cutting, welding, deburring, etc.) have to be performed along a given open-ended curve or a given closed contour. But the actual starting point for the task is usually irrelevant. This last fact is typically neglected in existing approaches for computing optimal paths. This limits the potential for optimization of the robot's movement a lot. Just think of the necessary paths in the example of Fig. 1, if the two effective tasks (2) and (4) would have been defined, such that both start at the same side of the workpiece (instead of opposite sides like depicted in Fig 1). Supporting task (3) would be much longer and so would be the overall movement of the welding.

Therefore, we propose to under-specify effective movements, by omitting precise definition of the starting/ending points. For example, for a closed contour, each point could be a starting point. Our research goal is to evaluate this idea and provide algorithms that will use these under-specified effective movements to automatically select the order and starting/ending points of the effective tasks as well as computing corresponding optimal support movements that will result in a (near-)optimal robotic movement. Optimality can be defined with respect to different metrics such as distance, time or energy consumption.

Although under-specification in effective tasks greatly increase the efficiency of optimization, another important flexibility comes from robot kinematics: multiplicity of inverse-kinematics solutions, multi-tool robot configuration and robot base location. For example, the costs of supporting movements usually heavily depend on the axis configuration[1] a robot uses for reaching any given starting/ending point of two subsequent effective tasks. In addition, real-life applications impose constraints that have to be considered: the collision constraint and partial order relations between robot tasks. All these additional sources of freedom and constraints will be involved in the optimization process.

The planned result of this PhD project – an algorithm to compute (near-)optimal robot trajectories by making use of under-specification – would allow for industrial robots to be used much more efficiently. The usage of industrial robots might become feasible in additional manufacturing processes where the costs of human labor is currently lower or equal to that of robots. It would reduce production costs, give a competitive advantage and would ultimately reduce consumer prices for the goods produced by robots.

## Related work

There exist many approaches on making robot movements more efficient. Specifically relevant for this proposal are approaches aiming at task sequence optimization, under-specification and collision-free planning. In Table 1 we summarized the mentioned approaches and listed whether they allow (a) sequence optimization, (b) collision-free planning, (c) under-specification, (d) pre-defined partial order, (e) multiple Inverse Kinematics solutions and/or (f) optimization of the robot base location.

One can observe that existing approaches mostly do not use flexibility from the under-specified task for optimization – although it promises enormous potential.

Summarizing, there is no approach that uses not only under-specification of the task for optimization, but also the extra freedom of the robot kinematics and constraints that come from requirements of the task. The goal of this research project is to develop such an approach.

## Objectives

Effective tasks are specified very precisely nowadays. However, many tasks often do not require a rigid specification, but allow for a certain degree of freedom. Therefore, the overall research goal of this project is to make use of this "freedom" for automatically computing optimal robotic movements. The best way to illustrate this idea is by showing a small example.

**Example:** Fig. 2 shows a real-world scenario from plastic manufacturing. This case study is inspired by an exemplary scenario based on a commercially available product [2] . The overall job for the robot is to cut a number of pieces out of a big plastic board and drill several holes. To do this, the robot is equipped with a multi-tool, which on one side consists of a cutting knife while a drill is mounted on the other end.

The left part of Fig. 2 shows the initial piece of plastic. No parts are cut out and the frame consists – due to the manufacturing process – of a rugged shape[3]. The whole manufacturing process involves 18 (effective) sub-tasks $T_1 ... T_{18}$. Namely cutting five closed contours ($T_1, T_4, T_{14}, T_{15}, T_{18}$), drilling 10 holes ($T_2, T_5 ... T_{13}$) and cutting three lines ($T_3, T_{16}, T_{17}$). The right part of Fig. 2 shows the workpiece after processing. All cuts have been made and all holes have been drilled.

Let's now assume, that paths for all cuts could be derived from a CAD model of the plastic board shown in Fig. 2. It seems like that an optimal sequence could be calculated by solving a traveling salesman problem (TSP), where the weight of each edge corresponds to the Euclidean distance. However, this is misleading, due to the fact that TSP input is a set of points, but not the contours, therefore the obtained tour will be suboptimal. We will illustrate this problem in more detail in Fig. 3. Here a more detailed view on the lower left part of the motivating example with only four effective tasks $T_1$, $T_2$, $T_3$ and $T_4$ is shown.

For each task $T_i$, potential starting and ending points have been defined as a tuple of points ($A_i$,$A'_i$). For the closed contours, these two points are (obviously) the same and we therefore omitted exit points ($A'_1$, $A'_2$ and $A'_4$) in Fig. 3 as they are identical to the entry points. The star symbol denotes the starting position of the robot end-effector.

These tuples are typically generated in/by the CAD system *without* taking any optimality of the robot program into account. As a matter of fact, these points often depend on *how* an engineer initially drew the construction sketch. Whenever a sub-component is re-used, it will be rotated or flipped (to for example allow production of a maximum number of parts with one pressing process). So in general these starting points are almost arbitrarily distributed along the contour.

Assume that all starting points were chosen as lower right corner. This is shown in the left part of Fig. 3. For this scenario an optimal schedule would be to start with $T_2$, then do line $T_3$, after that go to the rectangle segment $T_1$ and finally do closed contour $T_4$. The total (Euclidean) distance of supporting tasks would be 256 mm. Now take a look at the right

[1]In general, every 6D point (position and orientation) can be reached by a robot with eight different robotic configurations – for example elbow-up vs. elbow down)

[2]see `http://www.kuka-robotics.com/en/solutions/solutions\_search/L\_R148\_Deburring\_of\_plastic\_engine\_covers.htm`, accessed on February 7, 2013

[3]When melted plastic is pressed into a form, it often partially leaves the pressing brackets. This results in uneven, rough external contours.

| Approach | Sequence Optimization | Collision-free planning | Under-Specified Tasks | Partial Order | Multiple IK | Base location optimization |
|---|---|---|---|---|---|---|
| (Spitz and Requicha 2000) | yes | yes | no | no | no | no |
| (Wurll, Henrich, and Wörn 1999) | yes | yes | no | no | yes | no |
| (Gueta et al. 2008) | yes | yes* | no | no | yes | no |
| (Berenson, Srinivasa, and Kuffner 2011) | no | yes | partly | no | yes | yes |
| (Gentilini, Margot, and Shimada 2011) | yes ** | no | partly | no | no | no |
| (Pan, Li, and Klette 2010)# | no | no | partly | no | no | no |
| (Saha et al. 2006) | yes | yes | no | no | yes*** | no |
| (Zacharia and Aspragathos 2005) | yes | no | no | no | yes | no |
| (Baizid et al. 2010) | yes | no | no | no | yes | yes |
| (Bu, Liu, and Tan 2009) | yes | partly | no | no | yes | yes |
| (Elinas 2009)# | yes | yes | no | yes | no | no |
| Proposed project | yes | yes | yes | yes | yes | yes |

Table 1: Overview of related approaches

# – approaches are not directly related to robotics and do not scale well for robotic applications
* – application specific approaches
** – limited to a few number of goals
*** – except the cases when it is redundant, i.e., infinite number of solutions exist

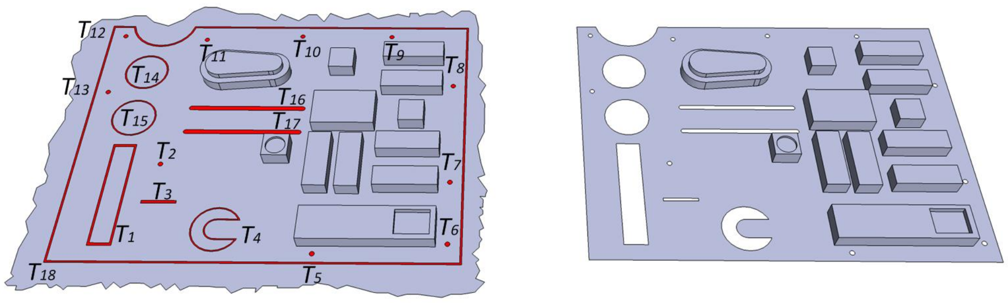

Figure 2: Complex composite task

part of Fig. 3. The starting position of the robot is still the same[4]. But different starting/ending points have been chosen. Now, another shortest path exists: $T_2$, $T_1$, $T_3$, $T_4$.

This path's length for supporting movements only accumulates to 134 mm. This example shows, that application of under-specification even in this simple case allows to reduce the path's length to 47.6% of the "naive" (TSP) optimization approach. For the sake of simplicity this example uses the Euclidean space for its calculations. In reality, the orientation of the end-effector should be included into the under-specified task description and will allow to obtain even better improvements in optimization.

To the best of our knowledge, there is no approach that could provide the solution of even this simple scenario in axis space.

Deeper analysis shows that effective tasks definitions are not the only sources of freedom. There are several other parameters that significantly influence the construction of the optimal sequence of the effective tasks:

- **Multi-tool robot configuration:** There is a freedom of altering between drilling and cutting operations in this industrial case. For example after drilling $T_{10}$ is done, the robot is free to choose the next task. It could choose the closest task (i.e., cutting line $T_{16}$) or stick with drilling and continue with the – further away – task $T_{11}$. Only in the first case, a tool switch is necessary (which typically increases the costs for the supporting movement a lot).
- **Multiple Inverse Kinematics solutions:** In most situations, there is no requirement to use certain solutions of inverse kinematics to perform cutting/drilling of contours/holes. In general a robot can reach every 6D position with eight different configurations (i.e., "elbow-up vs. elbow-down", "in-front vs. overhead" and "fore-hand vs. backhand"). However, which configurations is choosen will affect the cost of a supporting movement a lot.
- **Base location/Home position:** It is extremely important where the robot is located, as it significantly influences the sequence of the effective tasks. If the robot base was chosen "poorly", then all other efforts for optimization might not give valuable results at all. If for example the home

[4]This position can of course also be optimized. We just omit this here for keeping the example simple.

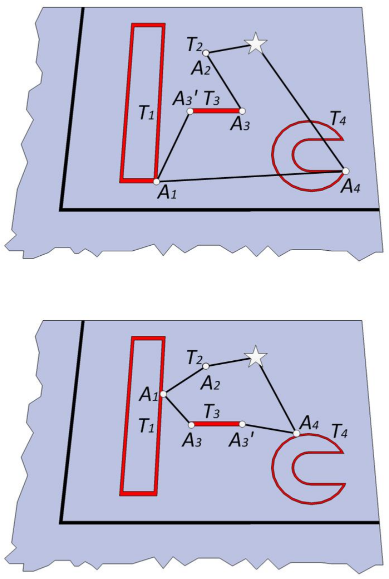

Figure 3: Complex composite task

position of the end-effector in the Fig. 3 would have been placed on the line between points $A_2$ and $A_4$ and the overall path length would be further reduced. Of course, this example is simplified and kinematics is ignored.

Besides these extra parameters and flexibilities, there also exist two important classes of restrictions, which must be taken into account:

- **Collision constraints:** It is obvious that a collision with obstacle may damage the robot. Therefore, collision-free planners must be integrated into the optimization process. Although there exist numerous algorithms for calculating collision-free paths, it will be very challenging to integrate these efficiently. The main reason is that these algorithms are computationally very expensive. While this might be practicable in simple planning scenarios, during an optimization run hundreds or thousands of such paths might have to be calculated.
- **Partial order:** Currently there are two extremes: either it is possible to strictly specify the sequence of effective tasks (conventional imperative programming) or omit all restrictions at all (current research optimization techniques). But many real world use cases require adherence to a partial order. Looking again at Fig 2, a partial ordering might exist, if tasks $T_{16}$ and $T_{17}$ would be welding tasks. In welding, a frequent requirement is not to do close weldings immediately after each other (because of thermal heating). This would place a restriction on the schedule by not allowing these two tasks execute immediately after each other with only a single supporting taks between them.

After our collaborations with various research and industrial partners we got a strong understanding that although under-specification could extremely change the optimization process by making it more efficient, it is still not applicable for real manufacturing scenarios. Real-life applications dictate extra industrial process requirements (partial order, collision avoidance) and extra freedom that come from robot kinematics and its position (multiple-tool support, multiple inverse kinematics solutions, base location). We believe, that integration of all these aspects in optimization is the key feature to obtain a real world sufficient approach.

The **main goal** of this doctoral project is to develop methods for computing optimal robotic movements by making use of (often implicit) under-specification. In particular, this leads to the following three subgoals:

**Subgoal 1:** Provide a formal (but also easy-to-use) framework for explicitly expressing under-specified effective and supporting tasks.

**Subgoal 2:** Develop algorithms that automatically compute optimal robotic movements for a given set of (under-specified) effective tasks taking into account industrial process constraints and extra flexibility that comes from robot kinematics.

**Subgoal 3:** Prove the increase in efficiency with a real-world scenario by integration into a standard offline-programming robotic framework.

## Conclusion

The overall aim of this doctoral project is to develop new methods for automatic generation of optimal robotic movements. We restrict ourselves to scenarios where the specification of effective computer-readable format and where no online planning (e.g., sensor integration etc.) is used.

The core idea is to make use of under-specification of effective tasks to open up potentials for optimization. We are also going to take into account robot flexibility (i.e., multiple solutions of inverse kinematics) as well as requirements for collision-free movements, because both heavily influence task sequencing and optimization.

## References

Baizid, K.; Chellali, R.; Yousnadj, A.; Meddahi, A.; and Bentaleb., T. 2010. Genetic algorithms based method for time optimization in robotized site. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1359–1364.

Berenson, D.; Srinivasa, S.; and Kuffner, J. 2011. Task space regions: A framework for pose-constrained manipulation planning. *International Journal of Robotics Research (IJRR)* 30(12):1435–1460.

Bu, W.; Liu, Z.; and Tan, J. 2009. Industrial robot layout based on operation sequence optimisation. *International Journal of Production Research* 41:4125–4145.

Elinas, P. 2009. Multi-goal planning for an autonomous blasthole drill. In Gerevini, A.; Howe, A. E.; Cesta, A.; and Refanidis, I., eds., *ICAPS*. AAAI.

Gentilini, I.; Margot, F.; and Shimada, K. 2011. The travelling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods and Software* 0:1–15.

Gueta, L. B.; Chiba, R.; Ota, J.; Ueyama, T.; and Arai, T. 2008. Coordinated motion control of a robot arm and a positioning table with arrangement of multiple goals. In *IEEE International Conference on Robotics and Automation, ICRA*, 2252–2258.

Pan, X.; Li, F.; and Klette, R. 2010. Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons. In *CCCG*, 175–178.

Saha, M.; Roughgarden, T.; Latombe, J.-C.; and Sánchez-Ante, G. 2006. Planning tours of robotic arms among partitioned goals. *Robotics Research* 25:207 – 223.

Spitz, S. N., and Requicha, A. A. G. 2000. Multiple-goals path planning for coordinate measuring machines. In *IEEE International Conference on Robotics and Automation*, 2322–2327.

Wurll, C.; Henrich, D.; and Wörn, H. 1999. Multi-Goal Path Planning for Industrial Robots. *International Conference on Robotics and Application (RA99)*.

Zacharia, P. T., and Aspragathos, N. A. 2005. Optimal robot task scheduling based on genetic algorithms. *Robotics and Computer-Integrated Manufacturing* 21:67–79.