THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Compositional optimization of large-scale discrete event systems

Fredrik Hagebring

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden, 2018

Compositional optimization of large-scale discrete event systems

Fredrik Hagebring

Technical Report No. R017/2018
ISSN 1403-266X
This thesis has been prepared using LaTeX.

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Göteborg, Sweden, December 2018

To my amazing wife, this is all thanks to you.

Abstract

Optimization of industrial processes such as manufacturing cells can have great impact on their performance. Finding optimal solutions to these large-scale systems is, however, a complex problem. They typically include multiple subsystems, and the search space generally grows exponentially with each subsystem. This is usually referred to as the *state explosion problem* and is a well-known problem within the control and optimization of automation systems.

This thesis proposes a new method of solving these optimization problems using a compositional optimization approach. This integrates optimization with techniques from compositional supervisory control, dividing the optimization of subsystems into separate sub-problems.

The key to this approach is the identification of local behavior in subsystems, behavior that is independent of all other subsystems. It is proven in this thesis that this local behavior can be optimized individually without affecting the global optimal solution. This is used by the approach, to reduce the state space in each subsystem, and then to utilize these reduced models compositionally when the global optimal solution is computed.

Results in this thesis show that compositional optimization efficiently can generate global optimal solutions to large-scale optimization problems, too big to solve based on traditional monolithic models. It is also shown that these techniques can be applied to several industrial applications, e.g. in logistics, manufacturing etc.

Keywords: Compositional Optimzation, Large-scale optimization, Automation, Discrete Event Systems, Discrete Optimization.

List of Publications

This thesis is based on the following publications:

[A] **F. Hagebring**, O. Wigström, B. Lennartson, S.I. Ware and R. Su, "Comparing MILP, CP, and A* for Multiple Stacker Crane Scheduling". *13th International Workshop on Discrete Event Systems (WODES)*, 2016, Xi'an, China

[B] **F. Hagebring** and B. Lennartson, "Compositional Optimzation of Discrete Event Systems". *14th IEEE Conference on Automation Science and Engineering (CASE)*, 2018, Munich, Germany.

[C] **F. Hagebring** and B. Lennartson, "Time-Optimal Control of Large-Scale Systems of Systems using Compositional Optimization". submitted for possible journal publication 2018.

Acronyms

MILP:	Mixed Integer Linear Programming
LP:	Linear Programming
CP:	Constraint Programming
SCT:	Supervisory Control Theory
CompOpt:	Compositional Optimization
DES:	Discrete Event System
DFA:	Deterministic Finite Automata
NFA:	Non-deterministic Finite Automata

Contents

Part I

Overview

CHAPTER 1

Introduction

Automation is the technology by which a process or procedure is performed with minimum human assistance [1]. This is something that we find everywhere in today's society, within manufacturing, logistics, communication and countless other areas. Moreover, the level of automation in these areas is continuously increasing to automate new tasks and improve the performance of the system. With an increased level of automation comes an increased complexity, where the systems include an increasing number of tasks and devices that should be coordinated.

To automate a single task can be tricky enough and to coordinate a process that includes hundreds of automated tasks is not a trivial problem. The control of such a process can, for simplicity, be described in two parts: (i) to ensure that all tasks are performed correctly and that the goal is finally achieved, and (ii) to maximize the efficiency of the whole process such that it can be performed as fast or as cheap as possible. The second part is an extension of the first one, which implies that it is no longer enough to reach the goal state. The controller has to be optimized such that the process can be done in the best possible way. This optimization is the main topic of this thesis.

The work presented in this thesis initially explores and compares different optimization paradigms for the specific type of optimization problems in Paper A and later presents a novel optimization technique specifically designed to excel at large-scale systems of systems in Paper B and C. The work focuses mainly on manufacturing and logistics system, where multiple robots collaborate to perform certain tasks. The same theories could, however, be applied to any automation system that is modeled similarly.

1.1 Research questions

The journey started 2015 when I was asked to help with an optimization of a logistics system, including three stacker cranes that should collaborate to perform a number of tasks. This sparked the first research question:

1. *Which optimization paradigms are suitable for large-scale production systems?*

This question was addressed in Paper A, which presents a comparison of three well known optimization paradigms; Mixed Integer Linear Programming (MILP), Constraint programming (CP) and graph search using A*, to explore their individual strengths and weaknesses for a generalization of the given problem.

The evaluation showed that all paradigms were able to solve the type of problem addressed, but none of them scaled well enough to handle larger systems. The reason was that the complexity of the problem scaled exponentially with the number of subsystems. The state space of the model became too big to be solved by any of the methods. It suffered from the well-known *state explosion problem*, also called the *curse of dimensionality* [2], [3], that is, as the number of state variables in the system increases, the size of the system state space grows exponentially.

To mitigate the state explosion I started to look for a viable method do decrease the complexity even before the optimization took part. Previous knowledge in supervisory control theory (SCT) lead to the next research question:

2. *Can SCT be integrated into the optimization to reduce the complexity of the problem?*

The idea was to use SCT to prune all in-feasible solutions such that they do not have to be considered during a subsequent optimization. SCT could probably reduce the complexity enough to solve slightly larger systems but it does suffer from the same state explosion problem and, hence, will have the same problem when dealing with large-scale systems. There are however techniques within SCT that can mitigate this problem to some extent by modular or compositional algorithms [4], when the system is separable into subsystems (system of systems). It has been shown in later work that compositional supervisory control can efficiently synthesize controllers for large-scale systems [5], [6]. In the third and final research question I search for a way to utilize the strength of the compositional SCT also within optimization:

3. *Can optimization of system of systems be done compositionally and still guarantee a global optimal solution?*

This final question became the inspiration to the novel optimization technique presented in Paper B and C, which is considered as the main contribution of this thesis.

1.2 Contributions

The main contribution presented in this thesis is the optimization technique, called *Compositional Optimization*, hereinafter called CompOpt. This method integrates techniques from compositional supervisory control with traditional graph based search algorithms. Its strength comes from the ability to reduce the state space of each subsystem individually by exploiting their local behavior, mitigating the state explosion that otherwise would occur during synchronization. Results in Paper B and C show that CompOpt drastically reduces the search space during the optimization of a realistic large-scale example and, hence, improves the computational complexity.

The most important components of this technique are: (i) a local optimization algorithm that computes a minimal reduction of each subsystem while maintaining the global optimal solution, (ii) an integrated synchronization and optimization algorithm that computes reduced synchronous compositions between subsystems, and (iii) the compositional integration of all sub-problems into a global optimal solution of the system without a monolithic model. Compositional methods are well known from SCT but, to the best of our knowledge,

using these in a general optimization formulation is a novel contribution.

1.3 Outline

This thesis is divided into two parts. The first part aims to give the reader an overview of the field of research and a better understanding of the concepts discussed in included papers. The included papers constitutes the second part.

The introduction provides the background and the research questions that have been the inspiration to the work. Chapter 2 includes a brief introduction to the most common concepts and paradigms within optimization of automation systems in general, and how to compare them. Chapter 3 focuses on the concept of modular/compositional optimization and specifically on positioning the main contributions of this thesis in relation to other work. A summary of the included papers is provided in Chapter 4. Concluding remarks and a direction of future work are provided in Chapter 5.

CHAPTER 2

Modelling, Control and Optimization of Automation Systems

An important tool in the development of automation systems is an unambiguous modelling paradigm that can be used to clearly specify the desired behavior and to model included components and their environment. These models and specifications can then be used in the control design or the optimization of such systems. The choice of a suitable modelling paradigm does, however, depend on the system that is being developed. This chapter gives a few guidelines into the vast research area of modelling, control and optimization of automation systems.

2.1 Discrete Event Systems

One of the most common ways to describe an automation system is as a Discrete Event System (DES) [7]. A DES can informally be defined as a discrete-state, event-driven system, which state evolution depends on the occurrence of instantaneous events that transition the system from one state to the next.

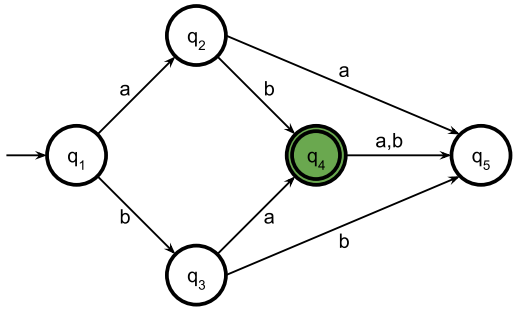

Figure 2.1: Visualization of as simple DFA model. The model specifies that the events a and b should be executed once each in arbitrary order. If any of them is executed more than once the system will reach an error state from which it can not return.

The work in thesis has mainly used automata to model DES, where the most common form of automata is probably *deterministic finite automata* (DFA). A DFA is defined by a 5-tuple $G := (Q, \Sigma, \delta, q_0, Q_m)$, where Q denotes the set of states, Σ denotes the set of events, $\delta : Q \times \Sigma \rightarrow Q$ denotes the transition function, $q_0 \in Q$ denotes the initial state and $Q_m \subseteq Q$ denotes the set of marked states. A DFA can be visualized as a graph, illustrated in Fig. 2.1, where nodes and edges in the graph represents states and transitions oif the DFA respectively. There are other similar modelling paradigms that also specialize in the modelling of DES, e.g. Petri nets, Markov chains etc.

Modelling an automation systems as s DFA allows for verification and synthesis using formal methods, such as *supervisory control theory* (SCT) [7]–[9], which both can be used to verify the correctness of an existing system as well as in controller design to ensure that he controller satisfy the desired behavior. Techniques related to SCT has been utilized in this thesis in connection with the compositional optimization, in Papers B and C, in order to prune away unfeasible solutions from the search space of the optimization.

2.2 Optimal Control

The control synthesis of SCT focus in general on the generation of maximally permissive controllers for a given set of specifications [7]. This means that

the purpose of the controller is to ensure that something bad never happens. This is useful when the plant is operated by an external controller or human operator. However, a controller of an automated task needs the ability to take *good decisions* when multiple paths exist, in order to eventually let the system reach a predefined goal state. Optimal control is the process of controlling the system not only to reach a goal state but to do it in the best way possible. This requires that the model is extended with a cost function that defines the notion of *good*. The aim of optimal control is to make the system reach the goal as cheap as possible, which constitutes an optimization problem or optimal control problem.

Cost function defines the notion of good

The cost function of an optimal control problem defines the cost of taking certain actions or executing specific events/transitions in case of a DES. The type of cost to include depends on the desired outcome of the optimization. In many applications this cost is represented by the execution time of the action, which will result in an optimization that minimizes the total execution time of the whole system, the makespan.

Constraints to specify required behavior

Just like in maximally permissive control synthesis, described in Section 2.1, there are some system constraints that are non-negotiable, such as: "State q must be reached eventually" or "Event σ must never occur". These are still viable in optimal control as specifications of desired behavior. The cost function defines how to choose between two alternatives, while the constraints define the ultimate goal.

Modeling and optimization paradigms

Modelling and solving optimal control problems is a vast area of research. There are countless different modeling and optimization paradigms, each of which have different strengths and weaknesses in terms of their ability and efficiency in solving different classes of problems. For example, if the optimal control problem can be solved using *Linear Programming* (LP) [3], then there are very efficient solvers that efficiently solve even the worst case problems.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 2.2: The well-known math puzzle, Sudoku. This can be seen as a combinatorial optimization problem and is usually solve using a similar technique as the propagation of constraints in CP.

An LP model is, however, restricted only to problems that can be expressed in canonical form as:

$$\begin{aligned} &\text{minimize} && \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b} \\ &\text{and} && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

where $\mathbf{x}$ represents the variables to optimize, $\mathbf{c}$ represents the costs related to each variable, A and $\mathbf{b}$ are a matrix and a vector of coefficients defining the constraints, and $\mathbf{c}^{\mathrm{T}}$ is the transpose of $\mathbf{c}$. The first line above corresponds to the objective of the model, where the variables in $\mathbf{x}$ should be chosen in such a way that the expression is minimized. The second line corresponds to the affine constraints, which, if combined, constitute the convex hull of a set of feasible solutions.

Another example is, *Constraint Programming* [10], which is especially efficient at certain type of combinatorial optimization problems, where parts of the state space can be pruned away quickly by propagating the constraints. An example of this is a Sudoku, shown in Fig. 2.2, which solution method is similar to the procedure of CP. The basic concept is to consider each variable individually and evaluate the set of feasible values of this variable, considering

each constraint individually. This process has to be iterated multiple times, until no changes to the variables can be made. For a Sudoku, this represents that the empty fields are considered one at a time. The feasible values for these fields are then evaluated based on the feasible values of all fields in the same row, the same column, and the same square. In each iteration there is at least one update to the set of feasible values, which will reveal additional information about other fields. In this way the propagation can continue until all fields have been assigned a value.

All classes of optimization problems can be modelled and solved using several different paradigms. Moreover, the choice of model and solver will greatly impact the computational complexity of the optimization. This is the basis of the first paper, Paper A, where we evaluate three of the most well-known optimization pardigms: Mixed Integer Linear Programming (MILP), Constraint Programming (CP) and Graph based search using A*.

CHAPTER 3

Compositional optimization

The aim of this chapter is to explain the concept of compositional optimization in general, the challenges and the potential it presents, and finally to give the reader a better understanding of CompOpt (the compositional optimization method presented in this thesis) through a set of examples that illustrate the key properties of the technique.

The main goal of compositional optimization is to minimize the state explosion problem for the optimization of systems consisting of multiple subsystems. The basic concept is to reduce each subsystem individually by exploiting *local* behaviour and then construct the global solution compositionally using their reduced models. This enables a global optimization of the complete automation system without considering the complete monolithic state space. The benefit is that this has the potential to reduce the overall complexity, since the computation of a monolithic model typically results in a state explosion, where the state space scales exponentially with the size and number of subsystems.

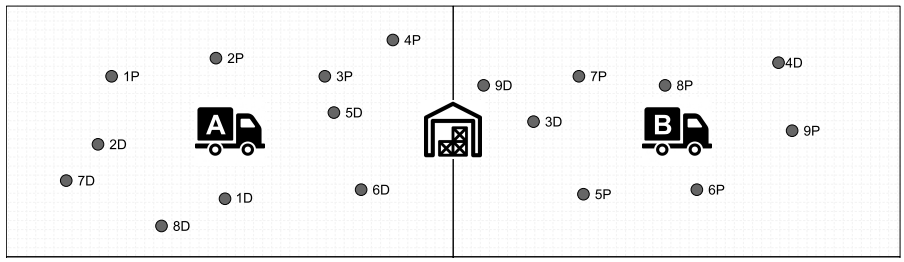

Figure 3.1: Illustration of a simple logistics system, consisting of two delivery trucks A and B, operating in adjacent neighbourhoods, that should pick up and deliver a total of nine packages. The pick up and delivery location of a package i is marked iP and iD respectively.

3.1 Motivating example

This section provides a motivating example to illustrate the impact of the state explosion problem and the potential benefit and challenges of using a compositional optimization approach. The example depicts a simple logistics system, consisting of two delivery trucks that pick up and deliver packages in separate zones. Every day, there are a list of packages that should be picked up and delivered within there operation area. One could argue that this motivating example does not really depicts the optimization of a large-scale system of systems, but the example is in fact already large enough for the purpose of this illustration. The system is illustrated in Fig. 3.1.

The figure shows the two trucks and there respective zone. In the center of the area there is a warehouse, which is where the trucks must start and end each day. The figure also includes an example of a scenario where nine packages should be picked up and delivered during the day. The pick up and delivery location of these packages are marked with dots on the map, where the labels iP and iD represent the pick up and delivery locations of package i respectively. Some packages should be picked up in one zone but delivered in another. In these cases the truck that picks up the package has to bring it back to the central warehouse where it can be moved over to the delivery truck. These type of switches between the trucks are assumed to occur only once a day. The objective in this example is to deliver all packages as quickly as possible, that is, the goal is to minimize the time when the last truck returns to the warehouse in the afternoon. The weights to be considered by the cost

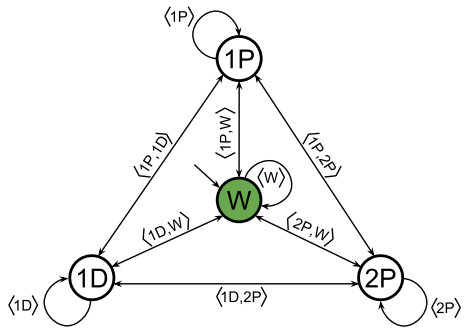

Figure 3.2: An automata model of the possible behavior of truck A, when assigned the tasks to pick up packages 1, 2 and deliver package 1. States represent the physical locations, while edges represent operations in these locations and travel in between. The state W represents the central warehouse, which is both the initial and the accepting state of the model.

function should in this case represent the time it takes to perform each task. The tasks include the pick up and delivery of packages, as well as the travel between these locations.

The physical position of each truck, can be modelled as a strongly connected graph, where nodes represent the locations of the warehouse and the pick up/delivery tasks, while the edges represent the travel in between. The actual pick up and delivery operations can be modelled as self loops in the nodes of the graph, indicating that a task is performed but the physical location does not change. In favor of readability, a reduced example where truck A only have to pick up and deliver package 1 and pick up package 2 is modelled using a simple automaton in Fig.3.2. The markings of the transitions are: (i) the self loops marked by $\langle x \rangle$ illustrating the different operations that can be performed in each location, including $\langle W \rangle$, which represents that the trucks switch packages at the central warehouse, and (ii) the edges between different locations marked by $\langle x, y \rangle$ representing the travel between two locations x and y. The central warehouse is marked green to illustrate that this is the desired goal state, the *accepting* state.

In addition to a model of the *possible* behavior, there are of course also models of the *desired* behavior. These are specified in Fig. 3.3. The specifi-

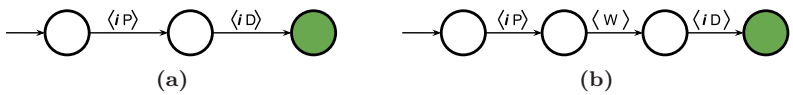

Figure 3.3: Generalized models of individual specifications for the route of each package. (a) applies to packages that is picked up and delivered by the same truck, (b) applies to packages that should be picked up by one truck and delivered by another.

cation in Fig. 3.3(a) is applied to all packages that should be picked up and delivered by the same truck. It specifies that the package has to be picked up and then delivered to its final delivery location exactly once. The specification in Fig. 3.3(b) is similar to 3.3(a) but should be applied whenever a package is to be picked up in one zone by truck X and delivered to another zone by truck Y. It is then required that the package is switched from one truck to the other in the central warehouse. Individual specifications like these have to be included for each package.

To evaluate the example, the scenario from Fig 3.1 is modelled as a system of systems, using plant models for each truck and specifications for each package to represent the subsystems, such as shown in Fig. 3.2 and Fig. 3.3. Any optimization applied using a monolithic approach would have to consider a search space spanning the complete synchronized behavior of all subsystems. This is true regardless of the optimization paradigm that is used. Advanced paradigms, such as MILP, CP, might be able to perform clever pruning of the search space in an early stage, but initially all possible combinations of states and transitions have to be considered. This is a potential problem since the size of the search space grows exponentially, due to the state explosion problem. The search space of the simple example shown here includes $342,144$ states and $6,329,115$ transitions, representing the synchronous composition of all subsystems.

When solving the same example using CompOpt, the optimization problem is partitioned into multiple sub-problems but the sum of states in the search spaces of all sub-problems combined only adds up to $16,396$ states. The reason that CompOpt is able to perform so much better than the monolithic approach is the ability to reduce the subsystems even before they are synchronized. The full search space is never computed, no unnecessary states have to be pruned away or evaluated. It is worth noting that CompOpt only represents one spe-

cific compositional approach, which most certainly can be further enhanced, but the purpose of this example is just to illustrate that there is much to gain from the ability to optimize systems of systems compositionally.

One could argue that there might exist more efficient models of this system than what is shown here. To a human it is for example obvious that the trucks can be partially optimized individually, since they drive in separate areas, have separate lists of tasks and so on. It is, however, not obvious exactly how this problem can be partitioned since there still exist dependencies between the trucks. Without digging into the details of exactly which tasks that can be considered local, there is no way to partition this problem manually. One benefit of using CompOpt is that it reduces the need of *smart* manual partitioning of the optimization problem, since it already exploits the local behavior maximally.

3.2 Challenges

Discrete optimization problems are considered NP-hard or NP-complete[11] in general and, hence, there cannot exist any general method that solves all of these in polynomial time. There is simply no way to completely solve the state explosion problem. To solve large-scale optimization problems one must instead exploit the properties of the problem. Efficient solution methods have been presented over the years for several classes of discrete optimization problems, such as traveling salesman, graph coloring, job-shop and minimum cut problems. All of these solution methods are, however, efficient mainly for the specific problem class that they are meant to address. When the specific problem at hand deviates from the standardized class the solution method typically becomes less efficient. The same is true also for compositional optimization, which exploits knowledge about the local behavior of subsystems to reduce the global state space.

Compositional SCT exploits the fact that if a certain sequence of events leads to something bad in one of the subsystems, then it is bad for the entire system and should be prevented. This makes sense since a correct execution of the full system requires all subsystems to function individually. In contrast to SCT, compositional optimization can not use such a simplification. There is no guarantee that the quickest or cheapest sequence of events in a subsystem is part of the quickest or cheapest sequence of events for the whole system.

Figure 3.4: A simple illustration of a railway system with two trains driving in opposite directions.

This can be illustrated with a simplified example of a railway system, shown in Fig 3.4. The example includes two trains A and B that are moving with the same speed in opposite directions from X to Y and from Y to X respectively, and the global objective is to minimize their combined traveling time. It is easy to see that the optimal solution requires train A to wait at Z until B has arrived, since B otherwise would not be able to start until A has finished. If we, however, only optimize the path of A, without taking B into consideration, it would be more efficient to just continue directly without delays. This illustrates one of the main limitations of compositional optimization. There is typically not enough information available to decide on a local optimal path for each subsystem individually. The key challenge then becomes the identification of those parts of the subsystems that we do have enough information about to reduce locally.

3.3 Previous Work

As mentioned in Section 2.2, many efficient methods for optimization of automation systems have been explored over the years, using a wide range of different optimization techniques [12]–[16]. Many of them have been proven efficient with respect to computational complexity, and typically scale polynomially with the size of the system. However, these methods normally do not utilize any modular or compositional approach and, hence, suffer greatly from the state explosion problem. It is not enough with methods that scale polynomially with the size of the system if the size itself scales exponentially with the number and size of its sub-systems.

In recent years, related work has been presented by other groups on modular or compositional methods, but these are either very restrictive in their reduction of the sub-systems or offer only approximative solutions [17], [18]. Particularly relevant to CompOpt is the work by Ware and Su in [19], which proposes a compositional method for synthesis of a time optimal controller.

In contrast to CompOpt, they do not provide any general local optimization algorithm that can reduce intermediate automata, unless these are completely disjoint. This is left for future research.

3.4 CompOpt – an Optimal Compositional Optimization Technique

Section 1.2 states that the main contribution of this thesis is CompOpt, developed in Papers B and C. The technique is specifically designed for the mitigation of the state explosion problem during the optimization of large-scale systems of systems. It is done by the identification and optimization of strictly local behaviour within each subsystem, combined with an integrated synchronization and optimization algorithm. These methods enable CompOpt to compute global optimal solutions without computing any monolithic models.

Local optimization reduces each subsystem individually

The local optimization method is the core of CompOpt. We prove in Paper B that this method can compute maximal reductions of each subsystem or sub-problem, called locally optimal reductions, without affecting the global optimal solution. The level of reduction is paramount to the complexity of the optimization, since it directly affects the extent of the state space growth in the subsequent synchronization.

The key to the local optimization method is to identify parts of the behavior in the subsystems that are strictly local, meaning that the behavior in these parts is independent of other subsystems. These parts can then be considered individually, without affecting the global problem.

To give a better understanding of the properties of local optimization, consider a small system consisting of two subsystems G_1, G_2. The task at hand is to optimize the first subsystem G_1 shown in Fig. 3.5(a), where the marking $\{\sigma, w\}$ of transitions indicates that the transition is activated by the event σ and has the weight w. The only available information of G_2 is that the event a is shared between the two subsystems in some way, how they interact is not revealed. The local optimization have to preserve the shared behavior, since this can affect the global behavior, which in this case means that the locally

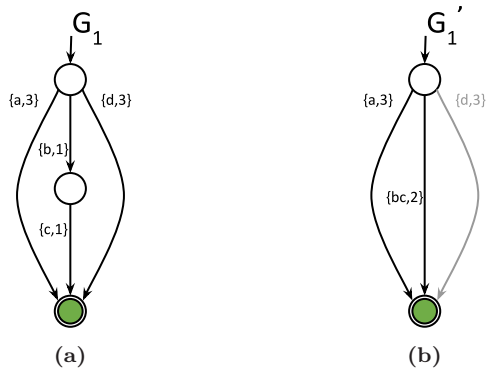

Figure 3.5: An illustration of the properties of local optimization. (a) shows a plant model of a subsystem G_1, where it is known that the event a is shared with another subsystem, (b) shows the locally optimal reduction of G_1, where the local transition $\{d, 3\}$ and the sequence $\{b, 1\}, \{c, 1\}$ has been replaced by an abstraction $\{bc, 2\}$.

optimal reduction of G_1, denoted G_1', will include the same transition over event a. The rest of the behavior can be considered local and can, hence, be optimized without affecting G_2. The resulting locally optimal reduction is shown in Fig. 3.5(b), where the event d is deactivated since it is more expansive than the sequence of local events b, c. The sequence of local events is abstracted to a single transition representing there combined behavior.

Integrated synchronization and optimization

The integrated synchronization and optimization method is the latest addition to CompOpt and was proposed for the first time in Paper C. This method mainly aims to reduce the complexity of dealing with a specific type of systems called time-weighted systems, where the cost function expresses the execution time of the tasks performed by each subsystem. CompOpt will then minimize the total execution time of the full behavior of the system, which requires synchronization of the time lines for the subsystems.

In SCT, these type of systems generally requires more complex modelling paradigms such as *timed automata* [20]. An alternative approach is to apply simplifications to the time-weighed system, e.g. discretization of the time

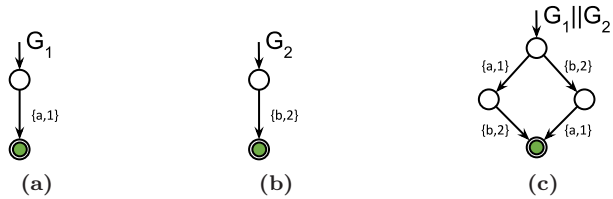

Figure 3.6: An example showing that default synchronous composition is insufficient when synchronizing time-weighted systems. (a-b) represent two independent subsystems that should run in parallel, (c) gives their synchronous composition.

line. This was applied using *tick automata* in Paper B. It was then sufficient to show the potential of CompOpt in general, but it was proved to be very inefficient, since it resulted in a reduced accuracy as well as a drastically decreased efficiency of the optimization.

The reason why these type of systems require a more complex modeling framework is that they no longer are pure DESs. The fact that the cost function or weights in these systems reflect the execution time of there transitions, makes it necessary to also consider the synchronization of the execution of these transitions, something that by construction can be neglected for an ordinary DES, since the transitions are assumed to occur instantaneously. The synchronous composition will, by construction, model parallel events as a sequence. This makes sense in a DES, since the events are by definition instantaneous, and, hence, can occur at the same time even if modelled sequentially. It does not make sense when the weights connected to the system transitions represent the execution time of the transitions, which would require a transition to finish its entire execution before it reaches the next state, where subsequent events can occur.

To illustrate this effect, consider the automata in Fig. 3.6, where (a) and (b) are two independent systems that run in parallel and (c) is the default synchronous composition of these systems. Let us first assume that the weight associated with each transition represents the energy consumption of the event. Then, a sequential model of the events a and b is correct. Each event, even if executed in parallel, still generates an individual energy consumption and the total consumption equals the sum of the two weights which will be the result

of the sequence of transitions in Fig. 3.6(c). Now let the weights represent the execution time of the transition. Since it is specified that the subsystems run in parallel, we expect G_1 and G_2 to reach its marked state in 1 and in 2 seconds respectively, independent of each other. Following the behavior of the synchronous composition in Fig. 3.6(c) does, however, indicate that it will be the sum of these execution times that is required to reach the marked state. This shows that the default synchronous composition known from DES is insufficient for the synchronization of time-weighted subsystems.

To fully understand the type of parallel behavior that should be considered, consider the *timed Petri net*[21] in Fig. 3.7. The parallel behavior of the events a and b can easily be incorporated using a Petri net model since this paradigm has the ability to use multiple tokens, which in this case are used to represent that the two transitions are performed in parallel. The timed Petri net model is however a more complex model than an automaton, including more information in each state about the current status of ongoing and possible transitions. To search through this system one would still need to explicitly search through each variation of these states, which would have the same effect as using a timed automaton or any other complex modelling paradigm.

The new approach, proposed in Paper C, for the synchronization of subsystems in CompOpt, is able to model the parallel execution of tasks in multiple time-weighted systems using only regular weighted automata, such as in Fig. 3.6. Moreover, by integrating an optimization heuristic, similar to the local optimization, in the synchronization it can utilize the weights of the transitions in order to compute a synchronous compositions that is partially reduced by construction. This enables the subsequent optimization of the composition to be faster due to the reduced search space.

To better understand the concept of the proposed approach, consider the weighted automaton in Fig. 3.8. This model represents the same synchronous composition as in Fig. 3.6 but has in this case been synchronized using the proposed approach in Paper C. The weights of the transitions of the synchronous composition $G_1 \parallel G_2$ are no longer directly related to the corresponding transitions in the subsystems G_1 and G_2. Instead they have been modified during the synchronization in order to represent the parallel execution of the events. One can see that the weight of transition a is now set to zero. This means that the the transition is performed instantaneous, just like a regular DES,

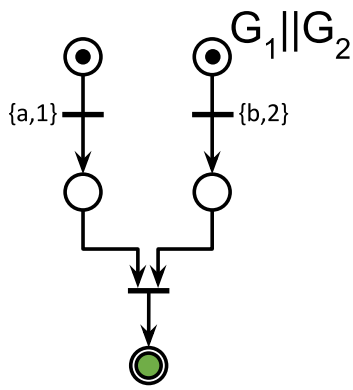

Figure 3.7: A timed Petri net model, representing a parallel execution of events a and b. Note that the final state is drawn as an accepting state only to highlight that it represents the same accepting state as in the corresponding automaton model. Generally, accepting states are not represented graphically for Petri nets.

and the actual execution time of a is instead covered during the subsequent transition. The weight of b is kept at 2 to allow both event a and event b to finish there execution. The system then ends up in the accepting state. This approach and further examples are explained in detail in Paper C.

Compositional computation of a global optimal solution

That CompOpt uses a compositional approach is pushed as one of the main contributions to this thesis. But once the local optimization and integrated synchronization and optimization is in place, the compositional computation actually becomes trivial.

From SCT we know that the compositional synthesis of a supervisor corresponds to an iterative process, where individual supervisors are computed for each subsystems and then combined incrementally while doing further synthesis step-wise. This is the case also in CompOpt. Let the set $G = \{G_1, G_2, \ldots, G_n\}$ be a weighted system that should be optimized using CompOpt. For example, let G_i' represent the locally optimal reduction of a subsys-

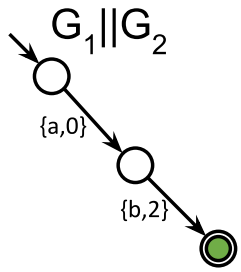

Figure 3.8: A time-weighted automaton representing the parallel execution of the events a and b.

tem G_i and let $G_i \parallel G_j$ be the composition of the systems G_i, G_j computed using the integrated synchronization and optimization method presented above, then the iterative process of CompOpt can be simplified to

$$S'_i = \left(G'_i \parallel S'_{i-1}\right)', \quad \forall i \in [1, n],$$

where S'_n will be the final global optimal solution.

The reason why this simple compositional approach is able to compute a global optimal solution is that each iteration of the algorithm expands on the local behavior by synchronizing additional subsystems. Once the final model S_n is reached, all tasks can be considered as local, which will let the locally optimal reduction S'_n represent the global optimal solution.

Strengths and limitations of the approach

The strength of CompOpt is first of all that it fully integrates the optimization into a compositional framework. It is shown in Papers B and C that this has the potential to drastically reduce the state space of the optimization problem without computing large monolithic models.

However, from Section 3.2 we remember that one of the main limitations to any compositional optimization approach is that the information available when considering a single subsystem typically is insufficient for the computation of a final optimal control strategy. CompOpt is no exception to this rule. It does guarantee that the local optimization computes a minimal reduction of each subsystem and that the final solution will be a global optima for

Table 3.1: Average number of states in the monolithic model of an instance with 2 robots when the number of tasks is increased from 2 up to 10, calculated using instances with ten different random seeds.

Tasks	States
2	8,140.0
3	15,392.0
4	96,030.8
5	180,032.0
6	912,585.6
7	1,638,088.0
8	6,986,696.0
9	11,981,064.0
10	42,450,952.0

the system, but we show in Papers B and C that the efficiency of CompOpt depends heavily on the complexity of the local behavior in the subsystems.

Taking the limitations into account, results in the papers show that CompOpt has the potential to improve efficiency of the optimization of large-scale systems of systems, when these have a complex local behavior in the subsystems. Tab. 3.1 includes a part of an example from Paper C, where a number of robots, here only 2, should each perform a varying number of tasks. The table indicates how the average size of the state space increases with the number of tasks to be performed. Without going into all the details about the example, these are described in fully in the paper, it is obvious that the exponential growth of this state space quickly becomes a problem for an optimization. In the paper, one can also observe that the growth becomes even worse when the number of robots are increased, instead of the number of tasks.

The results of solving the same systems as in Tab. 3.1 using CompOpt is shown in Fig. 3.9. Observe that number of states are represented using a logarithmic scale. The figure originally comes from Paper C and includes also varying number of robots but the bottom blue line of the graph represents the case with two robots, which is equivalent to the example in the table. One can see in the figure that the state space no longer grows exponentially with an increased number of tasks. Instead the growth resembles a low degree polynomial, since the line is almost linear. The paper further evaluates the

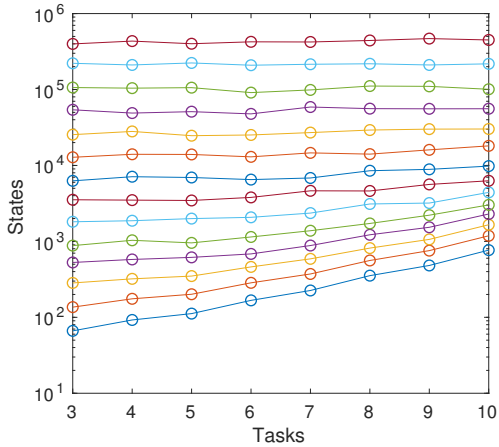

Figure 3.9: An example from Paper C, illustrating how the state space of CompOpt scales with the number of robots and tasks per robot. Each line in the plot represents a fixed number of robots ranging from 2 to 15.

potential of the approach and also includes detailed examples that illustrate possible limitations of the optimization method.

Applying compositional optimization in industry

This thesis gives examples of industrial applications both from logistics, in the motivating example of Section 3.1, and manufacturing industry, shown in the large-scale examples of Papers B and C. The general formulation of CompOpt does, however, enable it to optimize any system of systems as long as these can be modelled using weighted automata. These type of systems can be found in a wide range of applications and are in no way restricted to only the traditional areas of industrial automation.

In Papers B and C we apply CompOpt on an artificial example of a robot cell in production industry. This specific example may not be entirely realistic but there are similar scenarios in industry today. The example presented in these papers is a simplification of a respoting problem in a welding robot cell. Just like the example, the real scenario includes multiple robots that operate in parallel on the same product but from different angles. During

a production cycle there are specific events that affect all robots similarly, such as the assembly of one additional sub-part to the product. A few of the welding operations performed by the robots has to be performed while the assembly robot is still gripping the part while a majority of the operations can or has to be performed once the assembly robot has left the zone.

CHAPTER 4

Summary of included papers

This chapter provides a summary of the included papers.

4.1 Paper A

F. Hagebring, O. Wigström, B. Lennartson, S.I. Ware and R. Su
Comparing MILP, CP, and A* for Multiple Stacker Crane Scheduling
13th International Workshop on Discrete Event Systems (WODES), 2016, Xi'an, China

This paper presents an optimization model for a logistics system containing three stacker cranes that collaborate to perform a number of tasks. This model is then used to compare and evaluate the strengths and weaknesses of the well established optimization paradigms *Mixed Integer Linear Programming* and *Constraint Programming*, as well as the simpler graph based search strategy A*. The comparison includes their ability to solve problem instances of different size as well as their ability to find good approximations to the original problem where no global optimal solution can be found.

4.2 Paper B

F. Hagebring and B. Lennartson
Compositional Optimization of Discrete Event Systems
14th IEEE Conference on Automation Science and Engineering (CASE), 2018, Munich, Germany.

This paper presents CompOpt as a novel optimization technique that integrates techniques from compositional supervisory control with traditional graph based search algorithms. Its strength comes from the ability to reduce the state space of each subsystem individually by exploiting their local behavior, mitigating the state explosion that otherwise would occur during synchronization. The technique shows great potential in dealing with large-scale systems of systems.

4.3 Paper C

F. Hagebring and B. Lennartson
Time-Optimal Control of Large-Scale Systems of Systems using Compositional Optimization
Submitted for possible journal publication. 2018

This paper improves on CompOpt by proposing a novel and efficient synchronization method for time-weighted systems, called *reduced asynchronous synchronization* (RAS). This method is able to synchronize the parallel behaviour of time-weighted subsystems without adding any additional states or transitions to their models. The key is the integration of an optimization heuristic that, similarly to the local optimization, reduces the state space of the synchronous composition by removing non-optimal or redundant solutions, while maintaining the global optimal solution. We show in this paper that this further improves the efficiency of CompOpt by strengthening the mitigation of the state explosion problem.

CHAPTER 5

Concluding Remarks and Future Work

The work initially started as an evaluation of existing optimization paradigms, trying to identify efficient methods to deal with systems of systems. The evaluation did, however, result in the realization that all the evaluated methods suffered severely from the state explosion problem. This was, of course, expected since the problem is caused by the modelling rather than the optimization it self. This became the inspiration to the remainder of this thesis and the aim has since then been to push the boundaries of large-scale optimization of systems of systems.

The major contribution of this thesis is a novel approach to optimization of large scale systems of systems. This method, called compositional optimization, integrates time optimal control with methods from compositional supervisory control.

There are three key components in this method: (i) a local optimization technique that reduce the size of each sub-system individually to mitigate the state explosion problem, (ii) an integrated synchronization and optimization technique that synchronize the behavior of multiple subsystems and at the same time reduces the global state space using a fully integrated optimization heuristic and (iii) the compositional approach that computes the global opti-

mal solution of the complete system using the results from (i) and (ii). It is proven in this thesis that the proposed compositional optimization approach both maintains the global optimal solution of the system and computes a minimum reduction of each subsystem.

It is shown in the included papers that this method has the potential to be very efficient in large-scale optimization. Moreover, it is shown that it can scale very well with the number of sub-systems. This is especially true when the subsystems have a complex local behavior, something that in a monolithic optimization would cause an exponential growth of the search space. It is shown in this paper that the method can calculate globally optimal solutions for large-scale industrial applications. The focus has mainly been on automation systems, including examples of manufacturing and logistics systems. Yet, the general theories presented can be applied to any system of systems, as long as it can be modelled as a discrete event system.

In future work it would be interesting to apply this method as an online optimization method in real industrial applications. The main challenge that this presents is to model these industrial applications in an efficient way, such that their local behavior can be exploited maximally by the approach. Additionally, it would also be of interest to implement parallel computation of the sub-problems in compositional optimization and implement this as a cloud service to evaluate the potential of having scalable computational power.

References

[1] M. P. Groover, *Fundamentals of modern manufacturing : materials, processes and systems.* Englewood Cliffs, N.J. : Prentice Hall, cop. 1996., 1996, ISBN: 0-13-312182-8.

[2] A. Valmari, "The state explosion problem", in *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the Volumes Are Based on the Advanced Course on Petri Nets*, 1998.

[3] S. I. Gass and M. C. Fu, *Encyclopedia of Operations Research and Management Science, 2013 Ed.* Springer US, 2013.

[4] K. C. Wong and W. M. Wonham, "Modular control and coordination of discrete-event systems", *Discrete Event Dynamic Systems*, vol. 8, no. 3, pp. 247–297, Oct. 1998.

[5] H. Flordal and R. Malik, "Compositional verification in supervisory control", *SIAM Journal on Control and Optimization*, vol. 48, no. 3, pp. 1914–1938, 2009.

[6] S. Mohajerani, R. Malik, and M. Fabian, "A framework for compositional synthesis of modular nonblocking supervisors", *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 150–162, Jan. 2014.

[7] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems, 2nd Ed.* Springer Science & Business Media, 2008.

[8] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[9] ——, "The control of discrete event systems", *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[10] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence. Elsevier Science, 2006.

[11] J. van Leeuwen, Ed., *Handbook of Theoretical Computer Science (Vol. A): Algorithms and Complexity*. Cambridge, MA, USA: MIT Press, 1990, ISBN: 0-444-88071-2.

[12] K. M. Passino and P. J. Antsaklis, "On the optimal control of discrete event systems", in *Proceedings of the 28th IEEE Conference on Decision and Control*, Dec. 1989.

[13] B. A. Brandin and W. M. Wonham, "Supervisory control of timed discrete-event systems", *IEEE Transactions on Automatic Control*, vol. 39, no. 2, pp. 329–342, Feb. 1994.

[14] J. Huang and R. Kumar, "Optimal nonblocking directed control of discrete event systems", *IEEE Transactions on Automatic Control*, vol. 53, no. 7, pp. 1592–1603, Aug. 2008.

[15] A. Kobetski and M. Fabian, "Time-optimal coordination of flexible manufacturing systems using deterministic finite automata and mixed integer linear programming", *Discrete Event Dynamic Systems*, vol. 19, no. 3, pp. 287–315, Sep. 2009.

[16] F. Hagebring, O. Wigström, B. Lennartson, S. I. Ware, and R. Su, "Comparing MILP, CP, and A* for multiple stacker crane scheduling", in *13th International Workshop on Discrete Event Systems (WODES)*, May 2016, pp. 63–70.

[17] R. Hill and S. Lafortune, "Planning under abstraction within a supervisory control context", in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016.

[18] ——, "Scaling the formal synthesis of supervisory control software for multiple robot systems", in *2017 American Control Conference (ACC)*, May 2017.

[19] S. Ware and R. Su, "Time optimal synthesis based upon sequential abstraction and its application to cluster tools", *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 772–784, Apr. 2017.

[20] R. Alur and D. L. Dill, “A theory of timed automata”, *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994, ISSN: 0304-3975.

[21] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*. Jan. 2010.