AN AGENT-BASED FRAMEWORK FOR PROBLEM SOLVING IN SYMBIOTIC SIMULATION SYSTEMS

NANYANG TECHNOLOGICAL UNIVERSITY

AN AGENT-BASED FRAMEWORK FOR PROBLEM SOLVING IN SYMBIOTIC SIMULATION SYSTEMS

HEIKO AYDT

HEIKO AYDT

SCHOOL OF COMPUTER ENGINEERING

2011

2011

AN AGENT-BASED FRAMEWORK FOR PROBLEM SOLVING IN SYMBIOTIC SIMULATION SYSTEMS

HEIKO AYDT

HEIKO AYDT

School of Computer Engineering

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of Doctor of Philosophy

2011

Acknowledgements

This dissertation would not have been possible without the valuable input of many people. Some have had major impact on my work while others had little but nevertheless important influence. Some are probably not even aware of how they have influenced the way how I looked at things...

First and foremost I would like to thank my academic supervisor Professor Wentong Cai and my project supervisor during most of my time in PDCC, Professor Stephen John Turner. Both have provided me with guidance and continuous support during my time as PhD student. I thank them in particular for having found ways to keep me funded and thus enabled me to finish my dissertation after the IMSS project ended. At this point, I would also like to thank A*STAR for partially funding my research and the various lessons they taught me (unknowingly and unwillingly but nevertheless) while working on their project.

Many thanks is also due to Assistant Professor Malcolm Yoke Hean Low for his useful comments on optimisation issues; Dr. Peter Lendermann in his self-chosen role as the devil's advocate for his many critical comments that helped me to see things clearer; and Boon Ping Gan for his patience and time spent in discussions with me to clarify the many details regarding the semiconductor manufacturing showcase.

One of the major milestones of this work, the problem solver and a corresponding journal paper would not have been possible without the support and help of Associate Professor Ong Yew-Soon, who despite the fact that I was not his PhD student, always

had time to answer my questions and read the various drafts of our paper. In this context, many thanks is also due to Professor Rassul Ayani from KTH for his many constructive comments and useful ideas whenever he was in Singapore – and of course for the tennis matches.

I would also like to thank the many great people I met in PDCC. Shayan Shahand, for being a great friend, dive buddy, and best man on my wedding. His exceptional ability to be optimistic has been a much needed counterweight to my (sometimes over-boarding) pessimism. Mike, for his support, the many interesting conversations over a cup of coffee/tea, invaluable advice regarding all sorts of things – and the fact that, finally, I was not the only Ang Moh any longer. Irene, for her great support and help in fixing all sorts of things and restarting cluster nodes whenever the damn Windows Server decided to do what Microsoft products are infamous for.

Also, thanks are due to John Holland and the people from SFI (although they probably have no clue who I am). To the first, because he decided to give a lecture on Complex Adaptive Systems during a time when I happened to be in the need of some inspiration regarding automated problem solving. Evolutionary computing (only one topic he addressed in his lecture) turned out to be exactly what I was looking for. To the latter because of their highly inspiring insights on science and, in particular, interdisciplinary research.

Thanks also to those from whom I learned how *not* to do things.

My moral debts to the various people mentioned above is dwarfed by how much I am indebted to my wife, who had to deal with a grumpy me whenever things didn't work out the way they were supposed to, who always followed me to all the various duathlon, biathlon, triathlon, and marathon events (even if it meant to get up at 4 am in the morning), and who despite my flaws has always been (and still is) there for me. It's her to whom this thesis is dedicated to.

Abstract

Symbiotic simulation can be used for problem solving processes in the context of applications concerned with real-world physical systems. These kind of systems are often highly complex and exhibit non-linear behaviour which necessitates the use of simulation techniques to evaluate possible solutions to problems concerned with these systems. Problem solving can be described as an optimisation process concerned with the minimisation/maximisation of one or more objectives (expressed in terms of performance indicators). Optimisation requires the use of knowledge regarding the problem in order to effectively and efficiently solve it. Although black-box optimisation (i.e., without use of any knowledge regarding the problem) can be performed, it is easily outperformed by algorithms that incorporate domain knowledge. The more knowledge about a certain problem is incorporated into an algorithm, the more specialised the algorithm becomes. While specialisation generally improves the performance of an algorithm in solving a particular problem, it does so only at cost of decreasing re-usability of the algorithm for other problems. This is due to the implications of the no-free-lunch theorems. An autonomous problem solver agent which is meant to replace a human problem solver and automatically perform what-if analyses, needs to be able to solve different problems during its life span. This requires a flexible approach that does not statically hard-code information about a problem into the problem solving algorithm. Instead, it is necessary to dynamically incorporate problem-specific knowledge. In this dissertation we address this issue and establish a framework for constructing problem solver agents, based on

symbiotic simulation.

The approach in this dissertation is three-fold. *First*, we establish a theory on symbiotic simulation which also takes consideration of related work. As a result we propose a taxonomy on symbiotic simulations and introduce various classes of symbiotic simulation systems. Based on this taxonomy we can clearly define problem solving in symbiotic simulation and specify the what-if analysis process. *Second*, we argue for the use of evolutionary computing and propose our method for separating problem specific knowledge from the implementation of an evolutionary algorithm using an appropriate language. For this purpose, we introduce the evolutionary computing modelling language (ECML) and explain how it can be used to specify structural information about genotypes as well as processing information that can be exploited by the algorithm in order to process genotypes. With our approach of using a descriptive language and a meta algorithm that interprets the language we can effectively separate problem-specific knowledge from the implementation of the algorithm. This enables a problem solver agent to use problem-specific knowledge as it becomes available in order to perform situation-specific what-if analyses. Furthermore, we establish an agent-based framework that incorporates this method and provides various functional components that are needed to construct problem solver agents based on symbiotic simulation. Some of these components also facilitate the coupling between the simulation system and the physical system. *Third*, we apply our framework to two applications with real-world background. The first application is concerned with a decision making problem in the context of semiconductor manufacturing and the second is concerned with a model and state identification problem in the context of radiation detection.

With a flexible approach, such as the one proposed in this dissertation, it is possible to perform situation-specific what-if analyses. We compare the flexible situation-specific approach with the general case where no situation-dependent problem-specific knowledge

is incorporated. Experimental results show clearly the advantage of our approach. Although specialisation can help to improve the performance of an optimisation process, our results also show a decline in performance due to overspecialisation. This is the case when overly restricting the search space. As a result, the optimisation process is no longer capable of finding appropriate solutions to a situation-specific problem. These findings indicate the importance of carefully assessing the required degree of specialisation.

This page is intentionally left blank.

Contents

List of Tables

List of Figures

Glossary

C

Class A Symbiotic Simulation System a symbiotic simulation system where the what-if analysis is concerned with decision making problems.

Class B Symbiotic Simulation System a symbiotic simulation system where the what-if analysis is concerned with model identification problems.

Class C Symbiotic Simulation System a symbiotic simulation system where the what-if analysis is concerned with state identification problems.

Class D Symbiotic Simulation System a symbiotic simulation system where the what-if analysis is concerned with anomaly detection.

Class E Symbiotic Simulation System a symbiotic simulation system where the what-if analysis is concerned with forecasting.

F

Fab industry jargon referring to a semiconductor manufacturing plant.

O

On-line Simulation a type of simulation which is initialised and/or driven by real-time sensor data from a physical system.

P

Problem Solver Agent a symbiotic simulation-based agent that is capable of autonomously and automatically solving optimisation problems concerned with the physical system.

S

Scenario Space the space of all possible what-if scenarios in the context of a what-if analysis.

Solution Space the space of all possible solutions to a specific problem concerning a physical system.

Symbiotic Simulation a paradigm which refers to a close relationship between a simulation system and a physical system.

Symbiotic Simulation System a system in which a physical system and a simulation system (which simulates the physical system) are closely coupled.

W

What-if Analysis the act of evaluating alternative what-if scenarios by means of simulation.

What-if Analysis Process the algorithmic process of conducting a what-if analysis automatically.

Acronyms

A-C . . . . . actuator component

ACTM-C . . . . . action management component

COTS . . . . . commercial-off-the-shelf

DDDAS . . . . . dynamic data-driven application system

EA . . . . . evolutionary algorithm

ECML . . . . . evolutionary computing modelling language

GPU . . . . . graphics processing units

IAEA . . . . . International Atomic Energy Agency

JSP . . . . . job-shop scheduling problem

MEA . . . . . meta evolutionary algorithm

MKP . . . . . multi-dimensional knapsack problem

NFL . . . . . no free lunch

RDD . . . . . radiological dispersal device

S-C . . . . . sensor component

SCEM-C . . . . . scenario management component

SIMA-C . . . . . simulation analysis component

SIMM-C . . . . . simulation management component

UAV . . . . . unmanned aerial vehicle

UML . . . . . unified modelling language

WIP . . . . . work in progress

WIPER . . . . . wireless phone based emergency response system

WORC-C . . . . . workflow controller component

Symbols

Symbols which are consistently used throughout this dissertation, across multiple chapters, are listed below. Additional symbols, used in the context of a single chapter only, are not listed here but introduced and explained in their corresponding chapters.

Symbol	**Description**	**Section**
$S = S' \cup S''$	Complete state information S consists of known and unknown state information: S' and S'', respectively	2.3.1
S_t, B_t, C_t	State information S_t, behaviour information B_t, and control information C_t at time t, respectively	2.3.1
$\mathcal{M}$	Set consisting of all models for a physical system	2.3.2
$\mathcal{M}^R \subset \mathcal{M}$	Set of reference models for a physical system	2.3.2
$M \in \mathcal{M}$	A model of the physical system	2.3.2
$M_R \in \mathcal{M}^R$	A reference model of the physical system	2.3.2
PS	Set of model parameters associated with state variables in the physical system	2.3.2
PC	Set of model parameters associated with control variables in the physical system system	2.3.2
PE	Set of model parameters associated with uncontrollable variables in the physical system	2.3.2

Symbol	Description	Section
SS	Simulated state information object	2.3.2
SB	Simulated behaviour information object	2.3.2
SS	Time series of simulated state information objects	2.3.2
SB	Time series of simulated behaviour information objects	2.3.2
r	Simulation period	2.3.2
w	A what-if scenario	2.4
w_R	Reference what-if scenario	2.4.4
PI	Performance indicator	2.5
Ω	Objective(s)	2.5
Γ	Problem-specific Information	2.5
X	Solution space	2.5
$X_O \in X$	Set of optimal solutions	2.5
$x \in X$	A solution	2.5
W	Scenario space	2.5
$W_O \in W$	Set of optimal what-if scenarios	2.5

Symbol	Description	Section
f_{tr}	Triggering function for the WIA process	2.5
f_c	Comparison function for PI values	2.5
f_{wis}	Composer function for creating what-if scenarios	2.5
f_{sim}	Simulation function for evaluating what-if scenarios	2.5
f_{pi}	Simulation result analysis function for obtaining PI values	2.5
f_{opt}	Optimisation function for obtaining X_O	2.5
f_{pa}	Pre-WIA process function for specifying situation-dependent Γ and Ω	2.5
$\mathcal{G}$	Set of all genotype instances	3.3.1
$g \in \mathcal{G}$	A genotype instance	3.3.1
G	A genotype model	3.3.1
lb, ub, res	Lower bound, upper bound, and resolution of a numerical allele	3.3.1.1
E	An entity within a genotype model	3.3.1.2
R	A relationship within a genotype model	3.3.1.2
v_i	i-th optional parameter for various EA operators	3.3.2

Symbol	Description	Section
p_{var}	Probability that a particular variation method is performed	3.3.2.1
p_{ls}	Probability that a particular local search method is performed	3.3.2.2
p_a	Probability that a specific allele is selected	3.3.2.3
$f_{opt}^{MEA\alpha}$	Proof-of-concept implementation of f_{opt} using MEAα	3.4
L_i	Set of links corresponding to relationship R_i	3.4.1
f_e	Problem-specific evaluation function	3.5
PPI	Performance indicator information about the physical system, provided by sensors	4.3.1.1
f_s	Sensor function for retrieving S_t, B_t, and PPI_t	4.3.1.1
f_{as}	Action selection function used by the ACTM-C for filtering a decision	4.3.1.6
f_a	Actuator function used by the A-C for implementing a decision into the physical system	4.3.1.7

Chapter 1

Introduction

1.1 Background

Many physical systems that are of interest to scientists, engineers, and managers cannot be studied without the use of computer simulations due to their complexity. A computer simulation imitates the behaviour of a physical system by using an appropriate model of the physical system. Provided that the fidelity of the model is sufficient, simulations can be used to predict the behaviour of the physical system. Generally, it can be distinguished between three components of a simulation study: inputs, outputs, and the model that connects these two [37] . For example, consider the example of a manufacturing system. The behaviour of such a system (e.g., in terms of performance) depends not only on its design (e.g., number and types of machines) but also on the way how it is operated (e.g., scheduling policies that assign jobs to machines). A simulation study can be conducted in order to predict the expected performance (output) based on, say, a discrete event model of the physical system for a particular scheduling policy (input).

In the above mentioned example, the input as well as the model are known/available. The missing output can be determined by performing a simulation. Based on the three components of a simulation study (i.e., input, output, model), it can distinguished between three basic types of problems [37]: 1) in an optimisation problem the model as well as the desired output is known while the corresponding input is unknown; 2) in a

modelling or system identification problem the input as well as the corresponding output is known but not the model that would map the given input to the output; and 3) in a simulation problem the input as well as the model are known and the output has to be determined. The above mentioned manufacturing example thus belongs to the third category. Problems that belong to this category are arguably easier to solve than problems from the other two categories. All it takes to solve a simulation problem is to perform a simulation[1].

Problems belonging to the other two categories are generally more difficult to solve because there can be many possible solutions. For a simulation problem, we can assume that there is a specific output for a given model and input. This true even for stochastic models because the output follows a certain distribution. However, in case of an optimisation or model identification problem, there can be many possible solutions. For example, in case of a model identification problem, there could be many models that map some given inputs to corresponding outputs. Depending on the application, it might not be necessary to find all solutions. Although it might be sufficient to find at least one, it may not be possible to directly derive a solution. For example, training of neural networks (a model identification problem) can be done by using back-propagation. However, this explicitly requires knowledge about the internal structure of the model. Many physical systems are highly non-linear and deriving direct solutions to these problems is thus not feasible.

If deriving a direct solution is not possible, it is possible to search for solutions by evaluating a (potentially large) number of candidates until a solution is found. The process of using computer simulation to analyse candidate solutions is commonly known as what-if analysis: "a data-intensive simulation whose goal is to inspect the behavior of a complex system ... under some given hypotheses (called scenarios). More pragmatically,

[1]Performing a simulation may actually not be so easy if, for example, there is not enough computing power available because the model is too complex.

what-if analysis measures how changes in a set of independent variables impact on a set of dependent variables with reference to a given simulation model" [54]. It should be noted that simulation-based what-if analysis is a specialised form of the general idea of a what-if analysis for which simulation is not necessarily required. Many forms of the what-if analyses we encounter in our daily businesses do not use simulation. For example, an analytical what-if analysis, discussing seven what-if scenarios concerned with terrorist attacks, is presented in [139]. However, in the scope of this dissertation we are specifically concerned with simulation-based what-if analysis in a problem solving context.

In order to obtain reliable results from a what-if analysis, it is important to choose an appropriate level of abstraction (i.e., resolution) for the simulation model. A model with a low resolution may be sufficient for long-term simulations used to analyse the steady-state behaviour of a physical system. However, on smaller time scales, the behaviour of the physical system may be dominated by transient phenomena that require a higher model resolution. The issue of model abstraction has also been highlighted by Golfarelli et al. in [54]. They address this issue by proposing a methodology for developing a simulation model that can reliably predict the behaviour of a real business process and can thus be used to execute a what-if analysis. In addition, they also propose a supporting formalism that can be used to specify various parameters that can be modified by the user to realise different what-if analysis use-cases and corresponding what-if scenarios [53].

Short-term simulations can be used for various purposes. For example, in today's fast-changing business environments, companies need to continuously adapt and improve their business processes in order to stay competitive [39]. This requires the ability of decision makers to act quickly as operating conditions are continuously changing. In contrast to strategic decision making, which is performed less frequently and covers time periods of several months and years, operational decision making is concerned with continuous improvement and adaptation, which may only cover relatively short time periods, in

terms of hours and days. Performing what-if analyses based on short-term simulations faces a number of challenges. In addition to general problems, such as the choice of an appropriate level of abstraction for the simulation model, there are two issues that are specific to operational problem solving:

- *Need for high-fidelity simulations.* Appropriate values for model parameters may not be known at design time of the model. For long-term simulations it is often sufficient to make reasonable assumptions (e.g., by choosing appropriate distribution functions). However, for short-term simulations, this approach may not be sufficient. In addition, analysis of long-term simulations is usually concerned with the behaviour of the system once it has reached a steady state. This is not the case for short-term simulations which are concerned with the behaviour of the system in the near future. As opposed to steady-state simulations, there is no warm-up period and the simulation (ideally) predicts the behaviour of the physical system in the near-future with a very high degree of fidelity. Such a high-fidelity simulation not only requires a sufficiently realistic model that reflects all important components of the physical system, but it also requires real-time sensor data to initialise the simulation with the current state of the physical system, i.e., to warm-start the simulation.

- *Need for solving problems in a timely manner.* The time available to find suitable solutions is typically limited in real world physical systems. For example, in case the operational conditions of a manufacturing system are changing (e.g., because of a machine breakdown), then there is typically only a certain amount of time available to adapt to this new situation before more drastic and disruptive measures have to be taken (e.g., partial shut-down of manufacturing system) in order to handle the situation. In addition, it may not be possible to evaluate all possible what-if

scenarios. It is therefore necessary to apply an appropriate search strategy in form of an algorithm that selects promising scenarios for further evaluation and ignores less promising ones. This requires either an expert user, who can use knowledge about the physical system and personal experience with similar situations in the past, or a suitable search heuristic.

To summarise, what-if analysis has to be performed on-line (i.e., it has to use sensor data) as state information of the physical system may have to be incorporated in order to obtain a high-fidelity model. In addition, it has to be performed in real-time, i.e., it has to finish within a certain period of time. Otherwise it would be too late for the solution to be implemented in the physical system. Manually performing a what-if analysis requires a human operator to perform the various tasks described above, i.e., the human operator needs to initialise simulations with sensor data and think about promising scenarios to simulate. In addition, the human operator would also have to manually analyse the simulation results and decide upon the need to evaluate more scenarios. However, the on-line and real-time requirements make what-if analysis a difficult task to perform manually as operational conditions are continuously changing. An automated approach is thus desirable.

1.2 Motivation

A promising approach to automated what-if analysis is symbiotic simulation, a paradigm which refers to a close relationship between a simulation system and a physical system [44]. In this symbiotic relationship, the simulation system benefits from real-time measurements of the physical system, provided by corresponding sensors. The physical system, on the other side, may benefit from the effects of solutions found by the simulation system. An essential concept of symbiotic simulation is the evaluation of a number

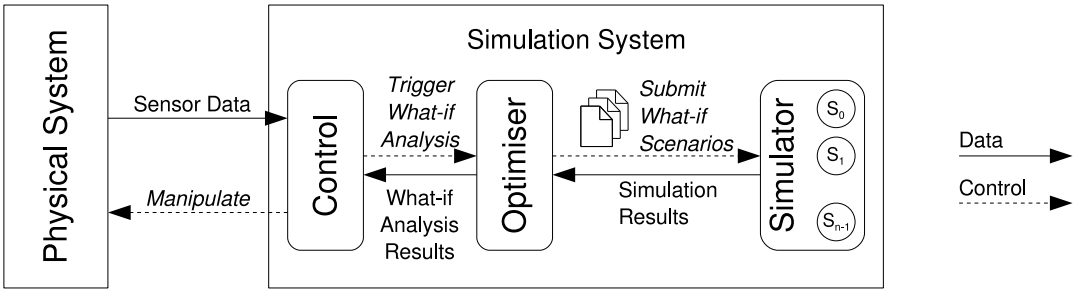


Figure 1.1: Overview of a symbiotic simulation system. A physical system is closely coupled with a simulation system. The simulation system observes the physical system and triggers a what-if analysis when needed. An optimiser is responsible for creating a number of what-if scenarios that are being simulated by a simulator.

of alternative what-if scenarios by means of simulation, further referred to as what-if analysis process (WIA Process). Essentially, the purpose of the WIA process is to find solutions to a problem regarding the physical system. In the context of decision making, for instance, the problem is to identify the best decision among a number of alternatives. Each alternative decision is represented by what-if scenario and the goal of the WIA process is to determine one or more decisions which can be expected to be most beneficial to the physical system. In Chapter 2 we will also discuss various kinds of problems in the context of symbiotic simulation, other than decision making.

In a symbiotic simulation system, the simulation system observes the physical system by means of sensors in real-time and automatically triggers the WIA process if necessary. This enables the simulation system to incorporate state information of the physical system to perform high-fidelity simulations and thus perform an on-line what-if analysis. A solution found by the WIA process can be directly implemented in the physical system by means of actuators (if available). If direct control over the physical system is not possible (i.e., if actuators are not available), then solutions are suggested to an external decision maker. Figure 1.1 illustrates an overview of a symbiotic simulation system.

Problem solving in the context of symbiotic simulation means finding solutions among a (potentially) large number of candidate solutions (each represented by a corresponding

what-if scenario). Given the real-time requirements explained above, an important issue is thus efficient scenario simulation. From a technical perspective, there are two distinct lines of multi-scenario simulation work in the literature of modelling & simulation. The first is multi-trajectory simulation [50, 51], a type of simulation which allows exploration of various alternative simulation trajectories, i.e., execution paths, concurrently. Multi-trajectory simulation relies on stochastic events to split alternative trajectories from the main simulation trajectory. The second is simulation cloning [59, 61], which aims at reducing redundancies between various simulation instances in order to reduce the need for computational resources and improve the performance. Simulation cloning has been developed with what-if analysis in mind and is based on the assumption that many what-if scenarios are, at least initially, equal or very similar. Both, multi-trajectory simulation and simulation cloning, also address the problem of having a very large number of scenarios that need to be simulated.

In multi-trajectory simulation the idea is to branch the original trajectory whenever a stochastic event is encountered, i.e., multi-trajectory resolution of events. This may lead to a large number of possible trajectories. If the number of trajectories is too large, it is not possible to explore all of them. Therefore, trajectory management is a major issue and aims to bound the maximum number of trajectories [50]. Gilmer and Sullivan briefly explain various techniques which can be used to limit resource consumption [51]. The first technique, referred to as "trajectory truncation", explicitly decides for each event whether to resolve it in multi-trajectory fashion or not. A second technique, which is only briefly mentioned, aims at reducing the number of states by consolidating states which are similar. Another technique, the "breadth first" approach aims at limiting the number of trajectories by allowing multi-trajectory resolution of events until a specified limit is reached.

The concept of simulation cloning was first introduced by Hybinette and Fujimoto [59, 61]. It is based on virtual logical processes [60] and performed in an incremental manner,

i.e., logical processes are only cloned if necessary. Decision points represent potential cloning points and, depending on possible outcomes of a decision, the further course of the simulation may be affected. Typically, different scenarios develop in different ways, i.e., they spread. Spreading means that the various simulation clones are very similar in the beginning and diverge from each other as time advances. Cloning is advantageous for applications that spread at a slow pace because redundancies (similarities) between scenarios can be exploited. At some point the simulation clones are entirely dissimilar, in which case there are no longer any redundancies that could be exploited to save computing time. Various extension have been described [1, 58] and cloning has also been applied to distributed simulations [19–21].

Depending on the application, the conduct of a what-if analysis can be very compute intensive due to the number of different what-if scenarios that have to be simulated. The simulation cloning approach by Hybinette and Fujimoto addresses this issue. The multi-trajectory approach by Gilmer and Sullivan also addresses this issue but to a lesser extent, i.e., the problem is addressed by essentially limiting exploration. Although both techniques address the issue of large scenario spaces, they do not fundamentally resolve it. In an interactive setting, a human user would manually decide on the set of what-if scenarios. However, in an automated setting, there is no interaction with a user. Hence, in order to adequately address this problem, an approach is needed that effectively limits the number of what-if scenarios that have to be simulated. This requires an appropriate search heuristic which is capable of selecting promising what-if scenarios and ignoring less promising ones.

In addition, developing application-specific symbiotic simulation systems is time consuming. Various functional components, required to perform an automated WIA process, have to be implemented as well as the infrastructure which facilitates the symbiotic relationship between the physical system and the simulation system. A framework for

symbiotic simulation systems can be expected to make it significantly easier for researchers and engineers to apply symbiotic simulation to their problem domain. The need for such a framework has already been articulated in [57]. This need is further justified by the variety of existing applications such as dynamic data-driven application system (DDDAS) [99] which have attracted the attention of many researchers in recent years. However, a general framework for symbiotic simulation system that also addresses the issues mentioned above does not exist yet and will thus also be the subject of this dissertation.

1.3 Problem Statement and Research Objectives

In a symbiotic simulation system, a human problem solver is replaced by an artificial problem solver agent that is interacting with the physical system (using sensors and actuators) in the same (or similar) way as the human problem solver would do. A fully automated problem solver agent would have to be autonomous, i.e., it would have to be capable of recognising and solving problems without the need for human intervention. This requires the ability to automatically perform the various activities that are typically performed manually by a human problem solver. Although autonomy may be desirable in many cases, it is not necessary to have a fully-automated problem solver agent. A semi-automated approach, where there is some interaction between the problem solver agent and a user, is also possible. For example, it may be more desirable to have a human user to analyse and specify the problem and leave the highly repetitive tasks of performing a what-if analysis to the problem solver agent.

Humans are capable of analysing a problem and exploiting domain-specific knowledge to solve problems. For example, this knowledge is possibly based on experience with similar problems in the past or in-depth knowledge about the underlying business process. Similarly to a human problem solver, a fully-automated problem solver agent

based on symbiotic simulation would also have to be able to incorporate problem-specific knowledge into the WIA process in order to effectively solve problems. The problems faced by the problem solver agent may change over time, i.e., subsequent what-if analyses may have to address different problems in the physical system. It is therefore necessary to specialise the problem solving algorithm, employed by the WIA process, to suit the problem at hand. As we will explain in greater detail in Chapter 2, many problems that are being considered in symbiotic simulation systems can be described as optimisation problems.

Specialisation is an important issue for solving optimisation problems. While general-purpose algorithms can be applied to a larger variety of problems, they are generally outperformed by algorithms that have been specifically designed to solve a particular problem. However, an algorithm that outperforms another algorithm for solving a particular class of problems will perform worse for solving at least one other class of problem. This insight is known as the no free lunch (NFL) theorems [132]. Essentially these theorems state that the average performance of an algorithm over all classes of problems remains constant. A performance advantage for a particular class of problems can only be achieved by accepting a degraded performance for at least one other class of problems. Although it has been shown that the NFL do not necessarily apply under certain circumstances [26, 133], their implications on the design of algorithms is well recognised. For example, the NFL theorems and their implications have been explained and discussed by Ho and Pepyne [55].

In addition to the need to incorporate problem-specific knowledge, there are also a number of symbiotic simulation-specific issues that have to be considered:

- *Real-time requirements.* The WIA process is a real-time optimisation process which has to be able to find a solution within a certain period of time, i.e., there is a deadline. This is due to the fact that solutions to a real-world problem may only

be meaningful for a certain period of time. This issue is aggravated by the fact that executing a simulation can be compute-intensive. In addition, many simulation models are stochastic and require the execution of multiple simulation replications in order to derive statistically sound results.

- *Optimal solution not required.* In the context of a real-world problem it is often more important to find a reasonably good solution in time, rather than an optimal solution. This is different from classical function optimisation where the goal is to find the global optimum, regardless whether it takes significantly more time to find a solution that is only marginally better than a previously found non-optimal solution. However, even if the optimal solution is not required, depending on the problem, finding a non-optimal (yet sufficiently good) solution may still be time intensive.

To summarise, an optimisation method for the WIA process has to be able to adapt to different situations by *dynamically* incorporating problem-specific knowledge. In addition, it has to be able to find a *sufficiently good* solution within a *limited period of time*. The objective of this research is to study symbiotic simulation systems and to propose a framework that allows the construction of application-specific problem solver agents that are capable of autonomously and automatically solving problems concerning the physical system by means of what-if analysis. Since the operational conditions of the physical system are dynamically changing, different what-if analyses may be concerned with different objectives and scenario spaces. Therefore, a problem solver agent needs to be able to analyse the current operational conditions and use domain knowledge about the problem in order to solve it.

1.4 Methodology

Before we proceed with explaining the methodology taken in this research, it is important to first explain where we see the place of symbiotic simulation in the context of the various fields and disciplines. Without doubt, symbiotic simulation has its roots in the modelling & simulation community that coined the term in the context of decision support applications [44]. However, the underlying concepts are not limited to the domain of modelling & simulation. With regards to the use of simulation in the context of decision making (in particular for operational decision making), modelling & simulation is overlapping with the field of operations research which is concerned with optimisation of real-world business processes. It is therefore not surprising that much work in the context of operations research is highly relevant to that of symbiotic simulation.

Optimisation is often based on mathematical models but with the need to solve real-world problems other methods, most notably meta heuristics such as the ones based on evolutionary computing, are used more and more because of various advantages [11, 40]. An important part of symbiotic simulation is optimisation as part of the WIA process. A corresponding optimisation method can use the ability of the symbiotic simulation system to evaluate various candidate solutions to a real-world problem by means of simulations. Problem solving is an important topic in the field of artificial intelligence and, to some extent, that of cognitive science and psychology which studies problem solving in humans. Among other topics, artificial intelligence also includes the study of evolutionary computing and meta heuristics. We see Symbiotic simulation not only at the intersection of modelling & simulation and operations research, but also at the intersection of all three fields. Figure 1.2 illustrates this view.

This dissertation is concerned with the development of a symbiotic simulation framework with a focus on automated problem solving. Due to the interdisciplinary nature of symbiotic simulation, this dissertation cannot be limited to the study of specific issues

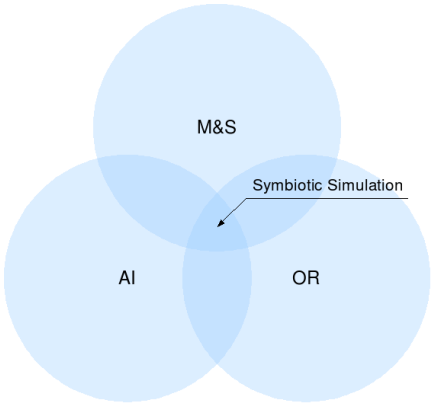


Figure 1.2: Symbiotic simulation at the intersection of modelling & simulation (M&S), artificial intelligence (AI), and operations research (OR).

of only one of these three fields. Instead, we take a holistic approach in this dissertation. This is also reflected in the methodology of the dissertation, consisting of three parts, each of which can be associated with one of the three fields:

1. *Theory on symbiotic simulation.* This part is probably most relevant to the modelling & simulation community as it aims at clarifying concepts of symbiotic simulation and establishing a taxonomy on symbiotic simulation. In addition, researchers of the modelling & simulation community are the primary audience for one of the major contributions of this dissertation: the symbiotic simulation framework.

2. *Problem Solving.* The focus of the dissertation is problem solving in the context of symbiotic simulation. Another major contribution of this dissertation is a problem solving approach which is more generally applicable and flexible than the existing ones. In addition, this approach is based on evolutionary computing methods and thus of particular interest to the artificial intelligence community.

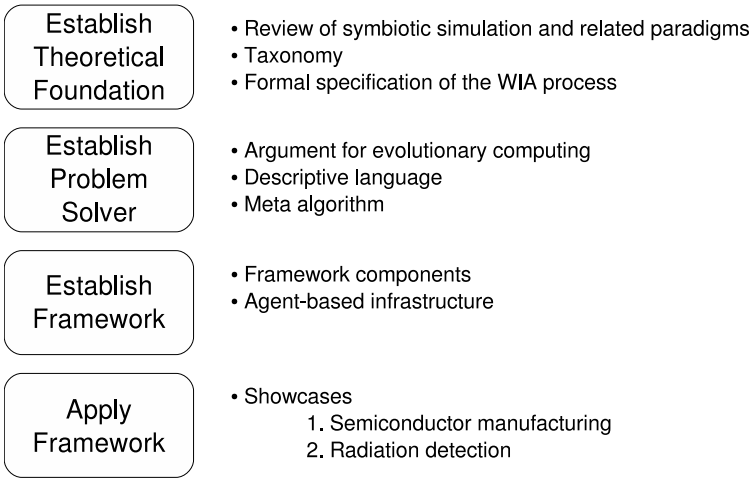


Figure 1.3: Overview of dissertation structure.

3. *Applications.* Based on the established theory and the introduced problem solving method, we propose a symbiotic simulation framework and apply it to two real-world problems in the context of semiconductor manufacturing and radiation detection. The applications of symbiotic simulation in order to solve real-world problems are highly relevant to the field of operations research.

1.5 Outline

The structure of the dissertation, illustrated in Figure 1.3, directly reflects the methodology. Due to the interdisciplinary nature of the dissertation, the relevant literature spans several areas. Therefore, instead of having a single literature review chapter, covering works from different areas, the dissertation is structured in such a way that facilitates a more natural flow of the argument. Hence, relevant literature is discussed in the various chapters.

In Chapter 2, we study the existing symbiotic simulation applications and classify the various kinds of symbiotic simulation systems, leading to a taxonomy of symbiotic simulation comprising of five distinct classes of symbiotic simulation systems. Three of

these classes are concerned with problem solving and thus subject to further investigation as part of this dissertation. We also analyse the workflow of a what-if analysis in the context of a symbiotic simulation system for problem solving and further formally specify the WIA process. This formal specification of the WIA process will serve as foundation for subsequent chapters. In addition, we also clarify ambiguities regarding terminology and give definitions for symbiotic simulation and on-line simulation, a highly related paradigm which is partially overlapping with symbiotic simulation.

In Chapter 3, we propose our approach for a flexible optimisation method, based on evolutionary computing. More specifically, we aim at separating domain-knowledge from the implementation of an algorithm by using a descriptive language. Furthermore, we introduce the concept of meta algorithms as a class of algorithms that make use of problem-specific information represented using a descriptive language such as the one proposed by us. This approach enables the problem solver agent to dynamically incorporate problem-specific knowledge. We apply our approach to a number of different standard optimisation problems to demonstrate the flexibility of the approach.

In Chapter 4, we propose a component-based architecture for the symbiotic simulation framework. The various components are based on the theoretical foundations established in Chapter 2. Furthermore, we discuss possible implementations of the framework and propose an agent-based infrastructure to realise the various functional components of the framework. We also describe how the optimisation method, developed in Chapter 3, is integrated into the framework. In addition, we discuss a number of conceptual applications.

In Chapter 5 and Chapter 6, we apply our framework to two proof-of-concept applications, concerned with semiconductor manufacturing and radiation detection, respectively. Each of these applications uses a different class of symbiotic simulation system. For each application, we describe the problem that needs to be solved and explain how problem-specific knowledge, obtained through analysing the physical system, can be dynamically

incorporated by using our descriptive language introduced in Chapter 3. We also discuss the application-specific design of the problem solver agent, based on the symbiotic simulation framework introduced in Chapter 4. The purpose of these applications is to demonstrate the flexibility of the optimisation method and the applicability of the framework in the context of real-world problems.

In Chapter 7 we repeat the problem statement and summarise our approach in order to solve the problem. In addition, we discuss a number of general issues and lessons that have been learned from applying our approach. Furthermore, we discuss future work.

Chapter 2

Theory on Symbiotic Simulation

2.1 Overview

In this chapter we establish the theoretical foundations necessary to develop a symbiotic simulation framework for automated problem solving. First, we discuss symbiotic simulation and related paradigms in Section 2.2. Then, we proceed by giving clear definitions of the various concepts that are relevant in the context of symbiotic simulation in Section 2.3. In particular, we explain the various components that are required to perform what-if analyses and derive a taxonomy of different classes of symbiotic simulation systems in Section 2.4. With this taxonomy we show that the original definition and scope of symbiotic simulation is too narrow and subsequently extend it. Different classes of symbiotic simulation systems are concerned with different types of problems. Not all of them are optimisation problems. For those classes that are concerned with solving optimisation problems, we state a general optimisation problem and formally specify the WIA process for optimisation-based problem solving in Section 2.5. This also serves as a basis for further narrowing the focus of the dissertation.

2.2 Symbiotic Simulation and Related Paradigms

Symbiosis has its origins in Biology. However, there is no universally agreed-upon definition and at the time of writing there are two established definitions. The first considers

mutualism, in which the relationship is beneficial for both partners, as the only form of symbiosis. The second considers symbiosis as it was originally intended by Anton de Bary who coined the term symbiosis in 1879 [30]. His definition is wider and considers several subcategories, including mutualism [32, 108]. Because there is no single definition of symbiosis, subcategories other than mutualism are often ignored [131]. This is presumably also the reason why mutualism is the only form of symbiosis which has been considered in the context of symbiotic simulations. Symbiotic simulation systems in the context of decision support and control can be considered mutually beneficial (later in Section 2.4 we will discuss symbiotic simulation systems without control feedback to the physical system). For example, this includes symbiotic simulation systems in semiconductor manufacturing [6, 83] and path planning for unmanned aerial vehicles (UAVs) [67, 85].

Related work on symbiotic simulation includes research on DDDAS, which is a paradigm closely related to that of symbiotic simulation and described in [99] as "a paradigm whereby application/simulations and measurements become a symbiotic feedback control system. DDDAS entails the ability to dynamically incorporate additional data into an executing application, and in reverse, the ability of an application to dynamically steer the measurement process". DDDAS emphasises the ability of the application or simulation to control and guide the measurement process. Yet another term which is related to DDDAS and symbiotic simulation is cyber-physical system (CPS) which refers to "the tight conjoining of and coordination between computational and physical resources" [100]. In a CPS, a physical system is tightly coupled with components from the cyber world (i.e., with computing and communication components). These cyber components monitor and control the physical system by using corresponding networks of sensors and actuators. CPS stresses the tight coupling of components from the cyber world and the physical world where cyber capabilities are deeply embedded in physical

systems [116]. The purpose of CPS is to significantly improve the capabilities (such as adaptability, autonomy, and safety to name a few) of physical systems. An important focus in CPS is thus the integration of various hardware and software components.

In general, all three paradigms (symbiotic simulation, DDDAS, and CPS) are highly related to each other and overlap thematically. A common feature of all three paradigms is their emphasis on a close relationship to a physical system by means of sensors and actuators. However, each paradigm has a slightly different focus. For example, CPS highly emphasises the importance of hardware and software integration as well as communication issues in distributed physical systems. As such, the focus of CPS is on the underlying technical infrastructure. DDDAS and symbiotic simulation, on the other hand, focus more on higher-level issues and implicitly assume that a technical infrastructure that facilitates the coupling between the application/simulation and the physical system is given. In particular, DDDAS appears to have a slightly more scientific focus where dynamic data-driven capabilities are primarily used to improve monitoring and modelling aspects by steering the measurement process. Symbiotic simulation is focusing on even higher level issues concerned with simulations of the physical system for various purposes. In addition, while both, DDDAS and CPS are thematic programmes by the National Science Foundation, symbiotic simulation is more precisely a technique that specifically focuses on modelling and simulation issues.

Another term which is often used synonymously for symbiotic simulation, is on-line simulation. For example, in [4] an on-line simulation is defined in the context of manufacturing as a "computerized system capable of performing both deterministic and stochastic simulations in real time (or quasi real time) to monitor, control, and schedule parts and resources in a discrete-part manufacturing environment". In a similar context, on-line simulation is used as part of an on-line planning and control system which uses on-line simulation to evaluate several scenarios concerned with control policies [29]. In both

cases, on-line simulation refers to a system which includes various components to monitor, simulate multiple scenarios, and to control the physical system. A slightly different definition of on-line simulation is given in [66], where it is described in the context of UAV path planning as a simulation that runs in real-time and in parallel with a physical system and does not necessarily include a feedback to the physical system. However, this definition of on-line simulation should not be confused with real-time simulation, as defined in [45], where advances in simulation time are paced by wallclock time.

2.3 Preliminaries

2.3.1 Physical System and Simulation System

A real-world physical system may consist of various sub-systems and may have various dependencies to other systems. Depending on the purpose of the what-if analysis, an appropriate model must include all relevant sub-systems and dependencies that are necessary to reflect the physical system with sufficient fidelity. For example, consider a manufacturing system which consists of a number of production resources (e.g., machines) and operators who make decision regarding the use of those production resources. Further, assume that for operational decision making we may only be interested in a particular group of machines. Now, if the purpose of the what-if analysis was to reduce the impact of maintenance activities on the throughput of the manufacturing system, then an appropriate model would have to include all those machines that are subject to maintenance activities. All other aspects about the manufacturing system can be ignored. However, if the purpose of the what-if analysis was to optimise a process schedule then it might be necessary to also consider customer orders and the expected arrival of required supplies. For the remainder of this dissertation, when we use the term physical system, we specifically refer to those parts of the real-world physical system that are of interest to the what-if analysis.

In a symbiotic simulation system, a physical system is coupled with a simulation system by sensors and actuators. The sensors provide information about the physical system and the actuators allow the simulation system to exercise control over the physical system. Some physical systems cannot be controlled and actuators are thus optional. We further distinguish between two types of information that is provided by sensors: information about the state S_t and behaviour B_t of the physical system at time t.

- *State information.* The state reflects the values of the various internal state variables of the physical system at any given time. We distinguish between known and unknown state information S' and S'', respectively. Depending on the physical system, it might be difficult or impossible to obtain the complete state $S = S' \cup S''$ of the physical system. For example, obtaining the exact state of the physical system may be too time or compute intensive and the state information may thus be outdated by the time it is available. In addition, while some state variables can be observed using corresponding sensors, this may not be possible with other state variables.

- *Behaviour information.* The behaviour reflects quantifiable information about a physical system that is not reflected by the state. For example, information regarding the behaviour (such as throughput of a manufacturing system) can be directly measured using corresponding sensors. In addition, behaviour information can be derived from observing the state of the physical system over a period of time. For example, throughput is not a state variable of a manufacturing system. Instead, it is derived from observations of consecutive states, i.e., the throughput can be derived by calculating the difference between corresponding state variables regarding work in progress (WIP) at different times.

The state information provided by the sensors may not be perfectly accurate. For many real-world physical systems this is certainly the case. For example, physical sensors (e.g., thermometers, pressure gauges, etc.) typically have a certain resolution, value range, and accuracy. In addition, if a network of sensors is used, then it is often necessary to aggregate and process the raw data in order to obtain data that can be used for simulation purposes. The issue of sensor data aggregation and processing is an important one and certainly deserves special attention and careful consideration when a symbiotic simulation is planned. However, data-mining and sensor-related issues are outside the scope of this dissertation. Here, we assume that sensor data is provided in a form that allows the simulation system to directly use without the need for further processing. This does not imply, however, that sensor data is assumed to be perfectly accurate. Instead, depending on the application context, we assume that certain information is available with a certain degree of accuracy.

A physical system can be controlled by receiving corresponding discrete control information C_t at time t. We consider control information to be discrete because decision making is typically a process which, possibly after some time of planning and reasoning, has a quantum of information (e.g., instructions) as outcome; and the outcome is meant to affect the physical system in a certain way. This information may only be valid for a limited period of time or invalidated by control information C_{t+1} which is propagated to the physical system at some later time $t + 1$. Figure 2.1 illustrates the links between the physical system and the simulation system.

For example, consider a semiconductor manufacturing application. The state S of the physical system may be comprised of information regarding the settings of the various machines as well as information regarding the number of WIP wafer lots, their current progress and whereabouts in the manufacturing system. What information exactly the state is comprised of depends on the application context and the specific model of the

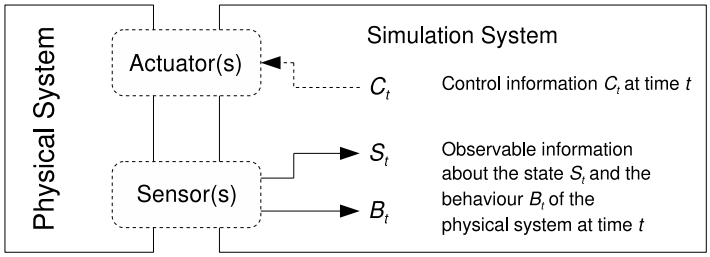


Figure 2.1: Overview of the link between the physical system and the simulation system in symbiotic simulation system.

physical system. However, it has to be sufficient information in order to initialise a simulation model and perform a high-fidelity simulation of the physical system. The behaviour B of the physical system in this example comprises of information regarding the current performance of the manufacturing process in terms of throughput and cycle times, for instance. Assume that decision making in this example is concerned with changing the configuration of the various machines. Control information C would thus be the instructions regarding the re-configuration of certain machines.

2.3.2 Simulation Model

Simulating the behaviour of the physical system requires an executable model. Many different models of the same physical system can be created, of which each emphasises different aspects of the physical system, possibly using different modelling techniques. A perfect model does not exist since all the models are always only an abstraction of the physical system. However, depending on the application, some models are more suitable than others to simulate a specific phenomenon concerned with the physical system – or, as George E. P. Box put it, "Essentially, all models are wrong, but some are useful" [12]. Therefore, for any physical system there can be multiple models, each describing either the same or different aspects and properties of the physical system. For example, assume that there are two models for describing the throughput of a manufacturing system.

Depending on the current operating conditions, one model might be more useful than the other. The set of models $\mathcal{M}$ is considered to contain all available models that can be used to simulate the physical system.

For simulation purposes a model $M \in \mathcal{M}$ has to be selected, which is capable of predicting the state and the behaviour of the physical system with sufficient fidelity. Depending on the purpose of the what-if analysis, some models might be more suitable than others. A model is suitable if it has sufficient predictive powers in order to produce useful results in the context of the what-if analysis. For example, if the purpose of the what-if analysis is concerned with the performance of a manufacturing system, then the model needs to be able to predict throughput, for instance, with sufficient accuracy. The subset of models that are suitable to simulate the physical system is further referred to as the set of reference models $\mathcal{M}^R \subset \mathcal{M}$. Whether a model $M \in \mathcal{M}^R$ also depends on time because as the operational conditions of the physical system are changing, the predictive power of one model may be affected. If this is the case, the current reference model $M_R \in \mathcal{M}^R$ has to be updated or a new reference model has to be found. The set of reference models is thus time dependent.

There are a number of parameters that are important in the context of symbiotic simulation and have to be explicitly considered. In a symbiotic simulation system, simulations are used to predict the behaviour of a physical system. The various simulation instances are initialised by using a specific state of the physical system. As for the initial state, it is possible to use the current state of the physical system. It is also possible to use any historic or synthesised state. Using existing state information to initialise the models avoids the need for a warm-up period for the simulation run. The state of the physical system may consist of a number of state variables and corresponding values. Therefore, a simulation model has to accept a number of state parameters that can be used to initialise a simulation with state information from the physical system. For this

purpose, the set PS of model parameters associated with state information is considered. The model parameters PS and state information from the physical system S are related, i.e., if the model is initialised with the current state of the physical system at time t, then $PS = S_t$.

Some physical systems can be controlled. Variables of the physical system that can be modified have to be reflected by the corresponding simulation model. For this purpose, the set of model parameters PC is considered. Note that there are physical systems that cannot be controlled from the perspective of the symbiotic simulation system. This may be due to the nature of the physical system (e.g., the atmosphere cannot be reasonably controlled yet) or due to policy (e.g., certain control variables of the physical system are intentionally not accessible to the simulation system for safety or security reasons). Therefore, depending on the application and the physical system, there may be no controllable variables, i.e., PC may be an empty set. This does not imply, however, that a physical system that cannot be controlled by the simulation system cannot be controlled by other means. The set PC of model parameters associated with controllable variables does explicitly refer to only those variables that are accessible to the simulation system. The model parameters PC and control information to the physical system C are related, i.e., if the same control information, that is used to perform a simulation, is used to actually control the physical system at time t, then $C_t = PC$.

A physical system exists within an environment which may, regardless whether it is natural or artificial, affect the physical system in some way. This external influence is not controllable from the simulation systems perspective (if it would be, it would be part of PC). Uncontrollable variables are typically sources of uncertainty and stochastic behaviour because it often has to be approximated by probabilistic assumptions. For example, incoming customer orders in a supply chain is an external source of influence that directly affects the supply chain. Predicting exact arrival times and numbers of customer

orders is impossible in most of cases and is therefore approximated by an appropriate probability distribution (e.g., exponential distribution) to model the interarrival times of customer orders. In order to simulate different what-if scenarios, assuming varying external influence, uncontrollable variables have to be explicitly considered by the simulation model. For this purpose, the set PE of model parameters associated with external influence is considered.

Given a sufficiently accurate model $M \in \mathcal{M}^R$ of the physical system and corresponding values for model parameters PS, PC, and PE, it is possible to predict (i.e., to simulate) the state and behaviour of the physical system in the near-future:

$$[\mathbf{SS}, \mathbf{SB}] = M(PS, PC, PE, r) \tag{2.1}$$

The result of the simulation consists of vectors $\mathbf{SS} = (SS_0, SS_1, \ldots)$ and $\mathbf{SB} = (SB_0, SB_1, \ldots)$, representing a time series of simulated state and behaviour information, respectively. Each vector has a length of $r + 1$ elements, where r denotes the simulation period. The first elements of these vectors represents the initial state and behaviour of the simulation run. All other elements represent the simulated (i.e., predicted) state and behaviour of the physical system at some time in the future. To further illustrate the relationship between state and behaviour information from the physical system (S and B) and simulated state and behaviour information (SS and SB), consider the example of a simulation that uses the current state S_t of the physical system at time t as initial state (i.e., $PS = S_t$) and $r = 2$. The simulation result for state information would thus be $\mathbf{SS} = (SS_t, SS_{t+1}, SS_{t+2})$ where $SS_t = S_t$ because the first element in the simulation result **SS** reflects the initial state (which is S_t). The same is not necessarily true for simulation results **SB** because behaviour information may depend on previous states (i.e., states at time $t-1, t-2, \ldots$) which may not available to the simulation, so it is possible that $SB_t = \varnothing$. For example, assume that the throughput of a manufacturing

system is calculated based on the difference of jobs between the current state SS_t at time t and a previous state SS_{t-1} at time $t-1$. When the simulation is started, only SS_t is available but not SS_{t-1}. Therefore, throughput for time t cannot be determine, i.e., $SB_t = \varnothing$.

Note that in this chapter we consider theoretical issues of symbiotic simulation. The complete simulation result $[\mathbf{SS}, \mathbf{SB}]$ is not necessarily required in practice. In particular, saving state information may not be desirable (e.g., due to space constraints or performance issues). Therefore, in practice it is possible that $\mathbf{SS} = \varnothing$. The same may apply to behaviour information (i.e., it may be that $\mathbf{SB} = \varnothing$).

2.4 Taxonomy

An important concept of symbiotic simulation is that of what-if analysis which entails the simulation for what-if scenarios. These simulations are used to predict the behaviour of the physical system depending on specific assumptions concerned with what-if questions regarding the physical system. For example, different what-if scenarios can be used to simulate the performance of a manufacturing system by assuming different combinations of machine settings (i.e., PC_0, PC_1, ...) and expected customer demand (i.e., PE_0, PE_1, ...) for certain products. A simulation of the physical system depends on the model and the various model parameters discussed above. Different scenarios are created by using different values for these elements. Therefore, a what-if scenario w is defined by a specific model M, specific values for the corresponding sets of model parameters PS, PC, PE, and a duration r that reflect the various assumptions for this scenario:

$$w = (M, PS, PC, PE, r) \tag{2.2}$$

What-if analysis is concerned with the evaluation of what-if scenarios, i.e., with mapping a what-if scenario w_i to a result $[\mathbf{SS}, \mathbf{SB}]_i$. Different what-if scenarios are created by

Table 2.1: Taxonomy of symbiotic simulation systems in terms of a model M, the set of state parameters PS, the set of control parameters PC, and the set of external influence parameters PE.

Class	M	PS	PC	PE	Purpose of What-if Analysis
A	fixed	fixed	variable	variable	Decision Support/Control
B	variable	fixed	variable	variable	Model Identification
C	fixed	variable	variable	variable	State Identification
D	fixed	fixed	fixed	fixed	Anomaly Detection
E	fixed	fixed	fixed	variable	Forecasting

using different values for the various sets of model parameters. For example, it is possible to use a fixed set of values for PS and PE for each what-if scenario but different values for PC. In this case, the what-if analysis is concerned with different decision making alternatives (e.g., $w_0 = (M, PS, PC_0, PE, r)$ and $w_1 = (M, PS, PC_1, PE, r)$, where PC_0 and PC_1 represent two different decision alternatives). A number of different classes of symbiotic simulation systems can be identified by considering the various sets of model parameters which are either fixed or variable. Each class of symbiotic simulation system is concerned with evaluating different values for the various elements of a what-if scenario, with the only exception being r which is not relevant for classification. An overview of the various classes of symbiotic simulation systems is illustrated in Table 2.1. The remainder of this section is concerned with a detailed explanation of each class.

2.4.1 Class A Symbiotic Simulation System

Class A symbiotic simulation systems consider the case where the problem is to identify optimal values for the parameters in set PC so that a desirable state and behaviour can be achieved in the physical system. This means that class A symbiotic simulation systems are concerned with decision making problems. For this purpose, the WIA process evaluates what-if scenarios that reflect different decision making alternatives. A particular alternative decision is represented by corresponding values for model parameters in

PC. In addition, what-if scenarios can also reflect different environmental conditions by assuming different values for the parameters in set PE. Generally, testing a particular decision under various environmental conditions increases the robustness of the decision making process. A decision that is less affected by environmental conditions are more robust and reliable as compared to a decision that has been tested under the same environmental conditions only. For example, assume that operational decision support in a semiconductor manufacturing system is concerned with decisions regarding the optimal configuration of the various machines. For this purpose, different machine configurations are reflected by different values for model parameters PC. External influence, such as customer order arrival which cannot be determined beforehand, may affect the performance of decisions. In order to assure the robustness of decision making, different scenarios with varying setup times can be considered by using corresponding values for PE.

The simulation system uses what-if analysis to investigate alternative decision making scenarios and the best decision is determined by analysing the simulation results. This decision is either proposed to an external decision maker or directly implemented in the physical system by means of actuators. We therefore further distinguish between a class A symbiotic simulation system for decision support or control, respectively. In case of a decision support system, control of the physical system is entirely up to the external decision maker which may or may not consider input from the symbiotic simulation system. Therefore, a decision support system only indirectly influences the physical system. A control system is the extension of a decision support system which is capable of directly implementing decisions by means of actuators. Although there may be significant differences between the realisation of these two kind of systems, from a conceptual point of view they are equivalent unless there is a probability that the external decision maker rejects a decision suggested by the simulation system. An overview of a class A symbiotic

simulation system for decision support or control is illustrated in Figure 2.2 and Figure 2.3, respectively.

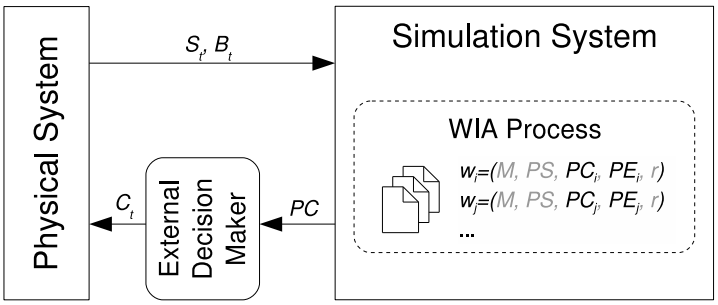


Figure 2.2: Overview of a class A symbiotic simulation system for decision support. The variable elements PC and PE of the what-if scenario are highlighted.

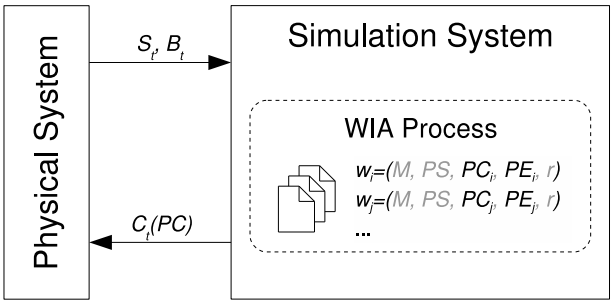


Figure 2.3: Overview of a class A symbiotic simulation system for control. The variable elements PC and PE of the what-if scenario are highlighted.

Many application examples of class A symbiotic simulation system for decision support can be found in the literature. For example, this includes answering what-if questions asked by fire fighters for situation assessment [97], simulation of alternative threat management scenarios upon an incident in a water distribution network [88], and dynamic path planning of a UAV using what-if scenarios [67]. In these examples, various scenarios are simulated and analysed to make appropriate decisions. There is also work in the literature which does not explicitly mention the use of what-if scenarios but does so implicitly. These include traffic control and management [46], and electric power transmission systems [93, 94]. Applications that directly implement decision into the physical

system (i.e., control systems) can be found in the literature as well. This includes the use of controlling agents to make necessary modifications to a semiconductor manufacturing system [83], on-line planning and control in manufacturing [29], implementation of simulated situational escalation scenarios by activating actuation mechanisms [98], and control of hydraulic gates in the context of flood water diversion [136].

2.4.2 Class B Symbiotic Simulation System

Class B symbiotic simulation system consider the case where a reference model is not available and has to be identified. This means, that class B symbiotic simulation system are concerned with model identification problems. For this purpose, the WIA process evaluates what-if scenarios that are concerned with different model alternatives. Each what-if scenario makes use of a different model and essentially represents an alternative hypothesis regarding the underlying causes of the state and behaviour of the physical system. The goal of a class B symbiotic simulation system is to determine a model that explains the state and behaviour of the physical system with sufficient accuracy.

We can further distinguish between active and passive model identification. Active model identification refers to the case where the simulation system is actively exercising influence on the physical system (using the set of control parameters PC) to test various hypotheses. In case of passive model identification, the simulation system is not exercising any influence on the physical system, i.e., PC is not used. As explained above, the set of model parameters PC refers only to those controllable variables of the physical system that can be influenced by the simulation system. Any external control of the physical system by a 3rd party would have to be reflected in the simulation model or as part of PE. An overview of an active or a passive class B symbiotic simulation system is illustrated in Figure 2.4 and Figure 2.5, respectively.

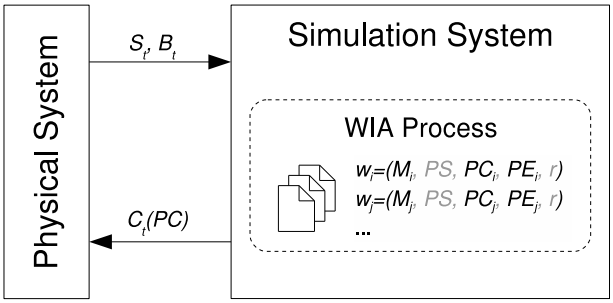


Figure 2.4: Overview of an active class B symbiotic simulation system. The variable elements M, PC, and PE of the what-if scenario are highlighted.

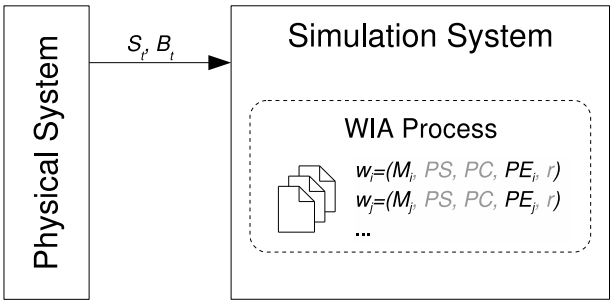


Figure 2.5: Overview of a passive class B symbiotic simulation system. The variable elements M and PE of the what-if scenario are highlighted.

An example for an active class B symbiotic simulation system can be found in [3], where it is used to characterise the three dimensional mechanical properties and geological structure of sites in seismically-active regions. In this application, an inverse wave propagation problem is solved using sensor data. Field experiments are performed to excite the soil at a particular location (PC). Measured ground motion data is collected and used to identify the material properties by solving the inverse wave propagation model. This model is used to describe how shock waves propagate through the soil. Solving the inverse problem is essentially concerned with the question regarding how the soil has to be characterised in order to explain the actually measured wave propagation. In addition, regions with high degree of uncertainty are identified and additional experiments are performed until a desired level of model accuracy has been achieved.

An example for a passive class B symbiotic simulation system is described in the context of WIPER, a wireless phone based emergency response system [86, 87]. This system collects data of communication and activity patterns from millions of cell phone users to detect and simulate emergency events as well as traffic jams. A detection and alert system is used to detect anomalies which indicate possible emergencies. Once detected, an appropriate model of the anomaly is determined by evaluating different hypotheses about the emergency events by means of simulation. Each hypothesis is reflected by different models. For example, a model may reflect a traffic accident situation. The simulation results are compared with sensor data in real time. The model which describes the event best is then used for forecasting the future course of the emergency event.

Evaluation of what-if scenarios, i.e., identification of the reference model, can be done by using historical sensor data from the physical system or real-time sensor data. In the first case, the purpose is to see whether the model is capable of predicting the current state and behaviour of the physical system with sufficient accuracy, based on a given model, a past initial state of the physical system, and recorded information regarding external influence. The second case initialises the simulation with the current state of the physical system and performs a real-time simulation where external influence is injected into the simulation as it becomes available. A combination of these two variants is also possible. Which method should be used, depends on the application context. The first method has the advantage of being able to perform a as-fast-as-possible simulation whereby the second method has to perform a real-time simulation which is synchronised with the physical system. Synchronisation may not be easy to realise and in some applications, only the first method is applicable. However, the first method makes use of historical data which may be too old and thus result in a reference model that is tailored to a past situation, i.e., the resulting reference model may be out of date.

2.4.3 Class C Symbiotic Simulation System

Class C symbiotic simulation systems consider the case where a reference model is available but the state of the physical system is unknown, or only partially known, and thus has to be identified. For example, incomplete state information might be a result of lack of sensor data. This means that class C symbiotic simulation system are concerned with state identification problems. For this purpose, the WIA process evaluates what-if scenarios that are concerned with different initial states. Class C symbiotic simulation systems are closely related to class B symbiotic simulation systems. The WIA process in both classes of symbiotic simulation systems is concerned with minimising the error between matching simulation results with data from the physical system. This is done by using what-if scenarios that are concerned with different hypotheses regarding the initial state or the model, respectively: class C symbiotic simulation system are concerned with identifying the set of state parameters PS and class B symbiotic simulation system are concerned with identification of M. Similarly to class B symbiotic simulation systems, it can be distinguished between an active and a passive class C symbiotic simulation system, depending on whether PC is used or not. An overview of an class C symbiotic simulation system is illustrated in Figure 2.6.

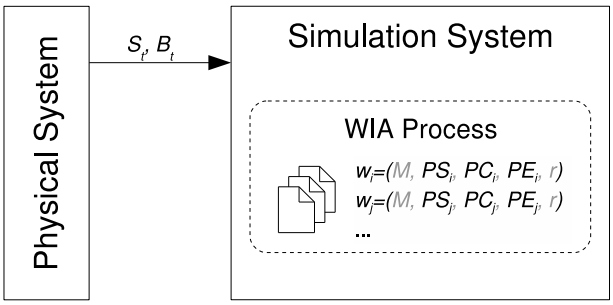


Figure 2.6: Overview of a class C symbiotic simulation system. The variable elements PS, PC, and PE of the what-if scenario is highlighted.

An example for a class C symbiotic simulation system is described in the context of

DDDAS where unknown boundary conditions in a fluid-thermal system (e.g., combustors, furnaces, and reactors) [76, 77] have to be determined. In this particular example, the average jet temperature and velocity are assumed unknown while only the jet pressure is assumed known, i.e., temperature and velocity cannot be directly measured. Instead measurements are taken from a restricted region in the flow system. Based on these measurements, a model of the system is used in an inverse procedure to determine the jet temperature and velocity. For this purpose, the model of the flow system is executed with different values regarding the jet temperature and velocity. For each set of parameters, the measured data is predicted and compared with the actual sensor measurements. These parameters are optimised until the error in the prediction of measured data is minimised. The results of this optimisation process is a simulation instance which can be used to predict the entire state of the physical system (i.e., the temperature and the velocity of the jet stream in the physical system).

2.4.4 Class D Symbiotic Simulation System

Class D symbiotic simulation systems consider the case where a reference model of the physical system is available and executed in parallel with the physical system by means of a real-time simulation [45]. Unlike other classes of symbiotic simulation systems, there are no variables in a class D symbiotic simulation system. Since the reference model can be used to predict the future state and behaviour of the physical system with a high degree of accuracy, the simulated behaviour of the physical system is continuously compared with the actual behaviour of the physical system. Discrepancies between simulated and actual behaviour indicate an anomaly which is due to either a problem with the model or a problem with the physical system. Although a class D symbiotic simulation system can be used to detect anomalies, it does not distinguish whether an anomaly is due to an abnormal behaviour of the physical system or due to model inaccuracy. Therefore, the source of the anomaly has to be

identified by other means. Compared with other classes of symbiotic simulation systems, which consider multiple scenarios, a class D symbiotic simulation system is different because it uses a single scenario only, i.e., the reference scenario w_R. This scenario uses a reference model M_R and reflects the current state of the physical system. An overview of a class D symbiotic simulation system is illustrated in Figure 2.7.

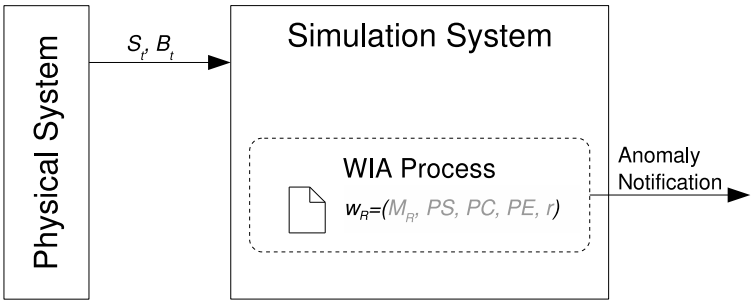


Figure 2.7: Overview of a class D symbiotic simulation system. There are no variable components in the reference scenario w_R.

For example, a class D symbiotic simulation system can be used to detect structural damage by comparing simulated and measured values, as described in [27]. An example of detecting an inaccurate model in the context of a social sciences application is described in [72]. Discrepancies between expected and actual behaviour are detected in the context of exception management on a shop floor in [68].

2.4.5 Class E Symbiotic Simulation System

Class E symbiotic simulation systems consider the case where a reference model of the physical system is available and the purpose is to create a high-fidelity forecast that incorporates real-time data from the physical system. The various what-if scenarios are concerned with different assumptions regarding external influence, i.e., the component of the what-if scenario that is variable is PE. The output of these simulations can be used by an external process for visualisation purposes or further analysis. A class E symbiotic

simulation system is similar to a class A symbiotic simulation system as it predicts future states of the physical system depending on a number of what-if scenarios. However, in contrast to a class A symbiotic simulation system, a class E symbiotic simulation system does not consider control influence by the simulation system. An overview of a class E symbiotic simulation system is illustrated in Figure 2.8.

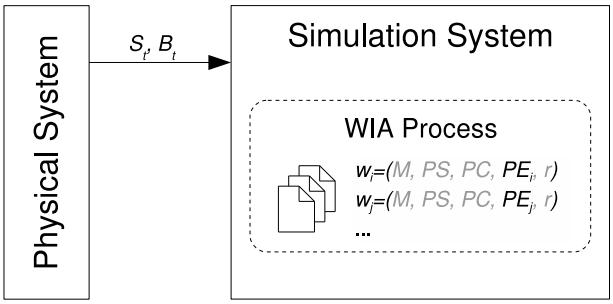


Figure 2.8: Overview of a class E symbiotic simulation system. The variables component PE of the what-if scenario is highlighted.

For example, a class E symbiotic simulation system can be applied to short-term wildland fire prediction [33, 90, 91] and image guided neurosurgery [89].

2.4.6 Hybrid Symbiotic Simulation Systems

Each of the different types of symbiotic simulation systems can be applied independently. For example, a single class A symbiotic simulation system is used in [83] to control semiconductor manufacturing back-end operations. However, depending on the application context it makes sense to combine various symbiotic simulation system. For example, the WIPER application [86] consists of several subsystems which are responsible for anomaly detection, simulation-assisted hypothesis testing (model identification), and decision support. Each of these subsystems could be realised with a symbiotic simulation system. A system which consists of a number of symbiotic simulation systems is further referred to as a hybrid symbiotic simulation system.

2.5 What-if Analysis Process

The various classes of symbiotic simulation systems can be divided into two groups, based on whether the purpose of the WIA process is optimisation-based problem solving or not.

- *Group 1* consists of class A, class B, and class C symbiotic simulation systems. These three classes of symbiotic simulation systems are concerned with problem solving based on optimisation. Class A symbiotic simulation systems are concerned with maximising the benefits for the physical system by finding an optimal or near optimal decision. Class B and class C symbiotic simulation systems are concerned with minimising the error between simulated and observed behaviour of the physical system by identifying the reference model or state of the physical system.

- *Group 2* consists of class D and class E symbiotic simulation systems. These two classes of symbiotic simulation systems are not concerned with optimisation. Class D symbiotic simulation systems are concerned with detection of anomalies either in the used simulation model or in the physical system by continuously comparing simulation and physical system. The problem here is to identify a discrepancy. Although this may not be trivial, it does not involve any form of optimisation (at least not on the level of the what-if analysis). Similarly, Class E symbiotic simulation systems are used to simulate a set of scenarios, using varying external influence, for the purpose of forecasting. This does not involve any form of optimisation from the perspective of the what-if analysis.

The symbiotic simulation system needs to be able to recognise a problem and trigger the WIA process. For this purpose, a corresponding triggering function $f_{tr}(S_t, B_t) = \{true, false\}$ is used. The problem may depend on the current operational conditions of the physical system and the symbiotic simulation system needs to be able to specify the

exact problem that needs to be solved before starting the WIA process. It is therefore important to have a clear notion of what comprises a problem in general and, more specifically, in the context of a particular application. According to Duncker, "a problem arises when a living creature has a goal but does not know how this goal is to be reached. Whenever one cannot go from the given situation to the desired situation simply by action, then there is the recourse to thinking ... Such thinking has the task of devising some action which may mediate between the existing and the desired situations" [35]. Similarly, Newell and Simon explained that "a person is confronted with a *problem* when he wants something and does not know immediately what series of actions he can perform to get it" [101].

Objective and Performance Indicator Generally, a problem involves an objective that has to be achieved (Duncker explicitly mentioned a goal and Newell and Simon do so implicitly by mentioning the desire for "something"). In the context of symbiotic simulation, we define an objective Ω in terms of some performance indicator PI. In class A symbiotic simulation systems the purpose of a what-if analysis is to find solutions to a decision making problem. Comparing the performance of alternative decisions requires a performance indicator PI which quantifies the performance of a simulated what-if scenario. In order to compare PI values, there has to be a comparison function $f_c(\Omega, PI_i, PI_j) = \{true, false\}$ which indicates whether a performance indicator value PI_i is better than or equal to another performance indicator value PI_j with respect to Ω. For example, a possible objective in the context of a semiconductor manufacturing system would thus be to maximise throughput: $\Omega = max\{PI\}$, where PI indicates the performance in terms of throughput. However, in class B and class C symbiotic simulation systems, the purpose of the what-if analyses is to find solutions to a model or state identification problem. It requires a comparison between the simulated physical

system and the real-world physical system. In this case, comparison is based on how well a what-if scenario (concerned with different models and/or assumptions regarding the state of the physical system) is capable of explaining the behaviour and state of the physical system. A performance indicator PI would thus reflect performance in terms of some measure of deviation. The objective would thus be $\Omega = min\{PI\}$.

Solution and Solution Space A problem is well-defined if there is a test which can be applied to a proposed solution in order to determine whether it is in fact a solution [95]. Furthermore, it is required that testing a solution has to be possible within a finite number of steps [95] (or "with a relatively small amount of processing effort" as described by Newell and Simon [101]). This definition of a well-defined problem is insufficient in the context of symbiotic simulation because the test only indicates whether a proposed solution actually is a solution or not, i.e., either yes or no. However, it does not allow to rank solutions, i.e., it implies that two solutions are equal. In practice this is often not sufficient. If, during the search process, many solutions are found then it must also be possible to rank them with respect to the quality of solutions. Therefore, it is important to be able to determine whether one solution is better or worse than another one. As a consequence, at the end of a what-if analysis, it is not only possible to distinguish between solutions and non-solutions, but also between good solutions and bad solutions. The best solutions found (there may be multiple solutions that are equal in terms of performance) represent the problem solver agent's answer to the problem.

In the context of real-world problems, solutions cannot always be represented by real numbers. Solutions in the context of symbiotic simulation cannot be limited to real values only. In addition, the solution space (i.e., the space of all solutions to a problem) and what exactly constitutes a solution is highly application-specific. For example, a solution in the context of a manufacturing problem is likely to be quite different from a solution

to a UAV path planning problem. While a solution of the first may include mappings of machines and setups, the latter may include a series of way points in a two dimensional space. Moreover, as explained in Section 1.3, an important issue with problem solving in symbiotic simulation systems is the fact that operational conditions are changing. As a consequence, the nature of the problem (and possible also the solution space) may change over time. Later in Chapter 3, we will introduce and discuss our approach, based on a descriptive language to define a problem-specific solution space. For now, it suffices to say that there is problem-specific information Γ which can be used to derive a solution space X from which candidate solutions $x \in X$ can be selected for further consideration.

In symbiotic simulation, solutions to a problem represent the corresponding values to only some of the elements of a what-if scenario. A solution is thus not sufficient to create a complete what-if scenario. For example, in case of a class A symbiotic simulation system, a solution represents the values for the model parameters PC. The other elements of a what-if scenario (i.e., M, PS, PE, r) need to be specified as well in order to evaluate the performance by means of simulation. For this purpose, a corresponding composer function $w_i = f_{wis}(x_i)$ is used which combines a solution x_i with the values of the other required elements to form complete what-if scenario w_i. Since the solution space is derived from Γ, a corresponding scenario space $W = \{f_{wis}(x_0), f_{wis}(x_1), \ldots\}$ can be derived by applying f_{wis} over all solutions in X.

A corresponding simulation function $[\mathbf{SS}, \mathbf{SB}]_i = f_{sim}(w_i)$ or $PI_i = f_{sim}(w_i)$ (depending on the application and class of symbiotic simulation system) is used to evaluate the what-if scenario. In practice, this function can be realised by using a commercial-off-the-shelf (COTS) simulation package, such as AutoSched[2] for example, or any other suitable simulation package that is capable of executing the simulation model specified by the what-if scenario. A simulation may either directly result in PI information or in state

[2] `http://www.appliedmaterials.com/services-software/library/autosched-ap`

and behaviour information $[\mathbf{SS}, \mathbf{SB}]$ which needs to be processed further to obtain PI information. For this purpose, an application-specific function $PI_i = f_{pi}([\mathbf{SS}, \mathbf{SB}]_i, S_t, B_t)$ or $PI_i = f_{pi}([\mathbf{SS}, \mathbf{SB}]_i)$ has to be used which calculates PI_i for a particular what-if scenario w_i based on the simulation results $[\mathbf{SS}, \mathbf{SB}]_i$. Depending on the class of symbiotic simulation system, this mapping function may also take the state S_t and behaviour B_t of the physical system at time t into account. While this is necessary for class B and class C symbiotic simulation systems, it is optional for a class A symbiotic simulation system because what-if scenarios can be evaluated based on the simulation results.

Searching for a Solution In symbiotic simulation, problem solving is done by the WIA process and can essentially be considered as simulation-based on-line optimisation. The general optimisation problem for problem solving in Group 1 symbiotic simulation systems can be stated as follows. The binary relation $\preceq$ is used to determine the partial order of what-if scenarios in the what-if scenario space $W = \{w_0, w_1, \ldots, w_{n-1}\}$ based on their associated performance indicators $PI_0, PI_1, \ldots PI_{n-1}$ by using the comparison function f_c. It holds that $w_i \preceq w_j$ if and only if $f_c(\Omega, PI_i, PI_j) = true$ (*i.e.*, if PI_i is better than or equal to PI_j with regards to objective Ω). It is assumed that for each problem, there is a non-empty set $X_o \in X$ of optimal solutions (and analogous to it, a set $W_o \in W$ of optimal what-if scenarios) for which the following holds:

$$\forall x_i, x_j \in X_o : x_i \preceq x_j \wedge x_j \preceq x_i \tag{2.3}$$

$$\forall x_i \in X_o, x_j \in X \backslash X_o : x_i \preceq x_j \tag{2.4}$$

This means that all solutions in X_o are equal in terms of their performance (Equation 2.3) and the solutions in X_o are better than the remaining individuals in X (Equation 2.4). Therefore, the objective of the search performed by the WIA process is to find X_o. The optimisation process is carried out by a suitable optimisation method, further denoted

by function $X_o = f_{opt}(\Gamma)$. The optimisation process relies on the the comparison function f_c which reflects the optimisation objectives Ω. In addition, the optimisation function f_{opt} relies on a notion of the solution space derived from problem-specific information Γ. For each what-if analysis that is carried out over the course of time, both the objectives Ω and the solution space may be different.

With respect to the problem solver, Newell and Simon noted that "the problem solver is not really given the set of possible solutions; instead he is given some process for generating elements of that set (all or some of them) in some order" [101]. This does also apply in the context of problem solving discussed in this dissertation. The problem solver has to be provided with problem-specific information Γ that allows the generation of solutions (i.e., what-if scenarios). This has to be done in a dynamic way because the problem, and thus the problem-specific information Γ for generating solutions, can be expected to be different for subsequent WIA processes, triggered at different times. The same applies for the objectives Ω, which may also change over time. For example, in a manufacturing environment, a performance problem caused by a tool break down may have to be handled differently than a performance problem caused by high load. Therefore, before the actual what-if analysis is carried out, there has to be a pre-WIA process which is responsible for analysing the physical system and to state the problem, i.e., the objectives Ω and the problem-specific information Γ required to generate solutions. For this purpose, a problem analysis function $[\Omega, \Gamma] = f_{pa}(S_t, B_t)$ is used.

Figure 2.9 illustrates the pre-WIA process and the WIA process.

2.6 Summary

In this chapter we have established a theoretical foundation of symbiotic simulation systems. In particular, we have identified five distinct classes of symbiotic simulation systems of which three of them are concerned with problem solving by means of optimisation.

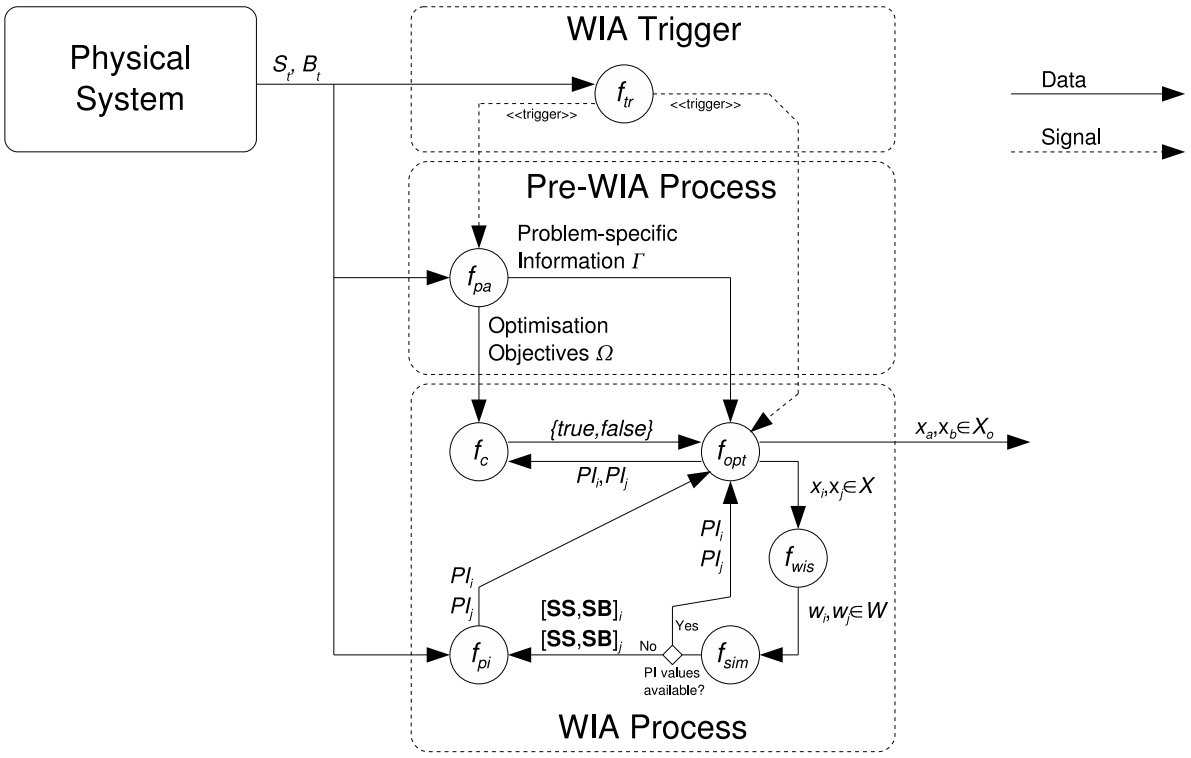


Figure 2.9: Overview of the Pre-WIA process and the WIA process.

In addition, some classes do not consider any form of control feedback to the physical system. This is in contrast with the original definition of symbiotic simulation [44] which emphasises a mutually beneficial relationship between the simulation system and the physical system. Mutualism refers to the case where both partners involved in the symbiosis benefit from each other. While the simulation system benefits from real-time measurements, the physical system benefits from control feedback created by the simulation system. However, the original definition based on mutualism is unnecessarily narrow as it rules out classes of symbiotic simulation systems other than decision support and control.

There is some disagreement in the community with respect to the terminology. What might be considered a symbiotic simulation by some may be referred to as on-line simulation by others and vice versa. On-line simulation has been used in various other works [34, 38, 110, 138]. However, none of them has provided a proper definition. Nevertheless, some of the concepts that have been described in the context of on-line simulation are also highly relevant to symbiotic simulation. This includes the simulation of multiple scenarios [4, 29, 110] and the focus on the physical system rather than the measurement process. In addition, real-time requirements mentioned in [66] are also important for symbiotic simulation. As consequence, both terms are often used interchangeably. Both terms have been used in the literature to refer to similar concepts. However, as explained in Section 2.2, none of them have been properly defined. In the context of this dissertation, we see the need for properly defining the terms "on-line simulation" and "symbiotic simulation". We define a on-line simulation as follows:

Definition 2.1 *An on-line simulation is a type of simulation which is initialised and/or driven by real-time sensor data from a physical system. An on-line simulation may be executed at any pace, i.e., paced to wallclock time, as fast as possible, or arbitrarily paced.*

In particular, an on-line simulation is concerned with a single scenario only, i.e., an on-line simulation does not simulate multiple scenarios as this would require some sort of scenario management mechanism.

In addition, we define an symbiotic simulation simulation as follows:

Definition 2.2 *Symbiotic simulation refers to the tight coupling of a simulation system and a physical system by means of sensors and actuators, facilitating real-time interaction. The physical system and the simulation system are inter-dependent and establish a symbiosis which is beneficial to at least one of them. The simulation system benefits from sensor data which is collected in real-time from the physical system and primarily used to initialise and drive on-line simulations about the physical system. The simulation system may employ an arbitrary number of what-if scenarios and on-line simulations which are concerned with the physical system. Symbiotic simulation thus includes a scenario management mechanism. The physical system may benefit from feedback by the simulation system (e.g., improved decision making and control). However, feedback from the simulation system to the physical system is optional.*

We have further divided the various classes of symbiotic simulation systems into two groups depending on whether the WIA process is concerned with optimisation-based problem solving. Although the purpose of the optimisation is different for each of three classes of symbiotic simulation systems in Group 1, the workflow of the WIA process, shown in Figure 2.9, is the same. We have described the WIA process and identified a number of functions that are required to perform a what-if analysis. In addition, we have stated a general optimisation problem. An important issue in this context, is the need for a pre-WIA process which is responsible for specifying the problem that has to be solved by the WIA process. A problem here is defined by an objective Ω and problem-specific information Γ that can be used to derive a solution space. The next two chapters

(Chapter 3 and Chapter 4) will be concerned with the details for problem solving (in particular with the representation of problem-specific information Γ) and the design of a framework that allows construction of application-specific symbiotic simulation systems.

As for the scope of this dissertation, we will focus on Group 1 symbiotic simulation systems only because the WIA process of symbiotic simulation systems in this group are concerned with optimisation problems. We have also distinguished between active and passive class B and C symbiotic simulation systems (depending on whether PC is used or not). In addition, we explained that, depending on the application, there can be external influence (expressed by PE). However, the focus of this dissertation is the WIA process and the problem of dynamically incorporating problem-specific knowledge. Considering variations of symbiotic simulation system, such as active class B symbiotic simulation system or class C symbiotic simulation system, as well as the impact of external influence on the robustness of the what-if analysis, are beyond the scope of this dissertation and thus subject to future work (see Section 7.3).

Chapter 3

Problem Solving

3.1 Overview

In the previous chapter we have established a general optimisation problem for Group 1 symbiotic simulation systems (i.e., class A to C symbiotic simulation systems). We have explained the need for objectives Ω and problem-specific information Γ, of which the latter is used to determine a solution space. A suitable solver can select candidate solutions from this solution space for further evaluation. In Section 1.3 we have explained that the nature of a problem may be different for consecutive what-if analyses. Furthermore, in the same section, we have also emphasised the importance of specialisation in the context of optimisation and problem solving. For this purpose, problem-specific information Γ is crucial as it can be used by a suitable algorithm, employed by the problem solver agent, to exhibit specialised behaviour.

This chapter is dedicated to problem solving in symbiotic simulation. In this chapter, we first discuss the background of problem solving in Section 3.2. This includes a discussion on various optimisation techniques in the context of real-world problem solving using symbiotic simulation which leads to an argument in favour of evolutionary algorithms (EAs). Furthermore, we discuss the importance of specialisation and explain different kinds of meta information for evolutionary algorithms (EAs). We then proceed by introducing a descriptive language that can be used to represent problem-specific information

Γ in Section 3.3. In addition, we introduce a corresponding meta algorithm that is capable of interpreting the language in Section 3.4. In Section 3.5 we apply our problem solving approach to standard optimisation problems from the literature in order to show the applicability of our approach. We conclude this chapter with a summary in Section 3.6.

The contribution of this chapter is two-fold. First, we introduce a high-level descriptive language that can be used to encode problem-specific information Γ. Second, we introduce a meta algorithm that is capable of interpreting this language. Later, in Chapter 4, we will integrate the problem solving approach described in this chapter into a symbiotic simulation framework.

3.2 Background

3.2.1 Advantages of Evolutionary Computing

When it comes to optimisation, an important issue is to select an appropriate optimisation method that suits the problem at hand. There is a large variety of mathematical optimisation methods and a correspondingly large body of literature on mathematical optimisation. For example, this includes convex optimisation, integer programming, linear programming, non-linear programming, quadratic programming, dynamic programming, stochastic programming, and various others. Mathematical optimisation is concerned with minimisation or maximisation of an objective function that has the following form: $f : \mathbb{R}^n \rightarrow \mathbb{R}$, where n is number of parameters. Depending on whether the objective function is known to have certain properties, a corresponding optimisation method can be used to solve it. For example, linear programming can only be applied if the objective function is linear and convex optimisation can be used if the shape of the objective function is known to be convex. Similarly, dynamic programming requires that a problem can be decomposed into sub-problems which are re-used several times throughout the

process. In addition, dynamic programming requires that an optimal solution can be constructed from the optimal solutions of its sub-problems.

The advantage of mathematical optimisation methods is that they are capable of finding exact solution. However, if a problem does not exhibit the various properties required by a specific method, this method cannot be applied. As explained by Bonissone et al. [11], many real-world problems exhibit properties, such as significant non-linearities, complex coupling, and multiple dimensions, which make it difficult, or even impossible, to use traditional approaches. In addition, many real-world problems cannot be easily described as continuous objective functions, neither can solutions to many real-world problems be described by a vector of real values as it is the case with mathematical functions. For example, the parameters may not be real values but representing discrete decisions. Furthermore, real-world problems are almost always stochastic due to uncertainty in the underlying real-world processes. While uncertainty is addressed by some of the mathematical approaches (e.g., stochastic and robust optimisation), they cannot be applied due to some of the issues mentioned. Symbiotic simulation is about solving problems in real-world physical systems and the underlying optimisation problems often exhibit properties which render mathematical optimisation methods unsuitable. Alternatives to mathematical approaches are thus required.

In contrast to mathematical optimisation methods, meta heuristics have several advantages [64]: they are not limited to continuous objective functions and are capable of handling discrete problems. The range of problems that can be solved by them is greater as compared to the conventional methods. For example, EAs, a popular type of meta heuristic, is increasingly used to solve real-world optimisation problems due to their advantages over mathematical methods [11]. This fact has also been acknowledged by Fogel [40] who explains that EAs offer considerable advantages for solving real-world problems. One advantage of EAs compared to mathematical programming-based algorithms is that they can be used to solve a large variety of problems regardless their

underlying structure. They do not impose similar constraints on the problem structure as it is the case with traditional optimisation methods such as linear or dynamic programming. Instead, EAs can be applied as long as it is possible to rank the individual solutions according to their fitness[3] in order to decide whether a particular solution is better than another one [40].

There is a variety of methods that can be classified as 'evolutionary algorithms'. Traditionally this includes genetic algorithms [56], evolutionary programming [41], genetic programming [78] and evolution strategies [117] but also more recent developments such as differential evolution [125]. Other commonly used meta heuristics which are also inspired by nature are swarm intelligence methods. Common swarm intelligence methods include particle swarm optimisation [36, 73], which is inspired by the social behaviour of bird flocking and fish schooling, ant colony optimisation [8], and bee colony optimisation [126]. Not all meta heuristics are necessarily inspired by nature (e.g., simulated annealing [16, 75], tabu search [52], and the cross-entropy method [119]). Which meta-heuristic should be used depends on the problem. However, while some heuristics are more generally applicable, such as EAs, others are more constrained. For example, although particle swarm optimisation methods have also been used to solve combinatorial optimisation problem (e.g., [63]), they have originally been developed for continuous optimisation problems. In this dissertation, we focus on the use of EAs because of their wide-spread use for solving combinatorial, continuous, and mixed problems.

An interesting feature of EAs is the so-called "any time behaviour" which refers to the fact that an EA can be stopped any time – hence the term – and the algorithm will have some solution available, even if it is not optimal [37]. Furthermore, typical progress of an

[3]The term fitness has its origins in Biology where it refers to an individuals ability to survive and reproduce. For example, physical strength is often an important contribution to an individuals overall fitness. In the context of evolutionary computing, the meaning of fitness is more abstract and usually defined with respect to the objectives of the EA. For example, if the objective is to minimise some cost function, then fitness is inversely proportional to that cost function.

EA shows that the fitness increase is significantly higher during the first half of the run as compared to the second half. Most of the progress, in terms of fitness improvement, is done during the earlier generations. With more and more generations being evaluated, it gets more and more difficult to further improve. EAs have reasonably good solutions available after only a few generations. This is an important advantage over mathematical optimisation methods which may not have a complete solution available when interrupted prematurely. Furthermore, Fogel points out that traditional optimisation procedures have to restart their calculations if any variable of the problem changes [40]. EAs can incorporate changes on-the-fly. This is an important advantage in the context of symbiotic simulation because real-time sensor data can be continuously injected into the optimisation process in order to quickly adapt to changing situations.

Despite the many advantages of EAs over traditional optimisation methods, they have disadvantages as well. The theory behind EAs is not fully established yet and there are not many guidelines regarding which EA configuration using what parameters is most suitable for a particular problem. In addition, unlike mathematical-programming approaches, EAs are not guaranteed to converge to the global optimum. However, as we already explained in Section 1.3, finding the optimal solution may not be important in the context of symbiotic simulation. EAs are particularly suitable in the context of symbiotic simulation where optimisation is concerned with real-world problems that exhibit complex behaviour, require real-time decision making and adaptability to dynamically changing operational conditions. Although mathematical optimisation algorithms may not be directly applicable, they can sometimes be used in combination with an EA. This allows combining the advantages from different optimisation methods. For example, a hybrid approach combining dynamic programming with an EA has been described in [137].

3.2.2 Importance of Specialisation

The importance of using domain knowledge in the design of effective EAs is widely acknowledged, not only as a consequence of the NFL theorems, but also as a result of experience with practical applications in this area. Specialisation is achieved mainly by embedding (i.e., hard-coding) available knowledge about a problem into the algorithms. An overview on the various ways to incorporate domain knowledge into EAs is described by Bonissone et al. [11]. Domain experts use their knowledge about a problem to choose a representation that ideally captures all the important features of the problem. In addition, they design specialised operators and choose parameters that are known to suit the problem at hand. This has led to a large variety of (sometimes highly) specialised EAs and meta-heuristics that have been developed to solve a particular problem. However, such a static approach to specialisation is not desirable in the context of symbiotic simulation because the problem solver agent will have to solve various kinds of problems throughout its life time and thus cannot be statically specialised to a particular kind of problem. Instead, an EA is required which can be specialised with respect to the requirements of a particular problem that has to be solved by the WIA process.

For example, different problems may require different operators and/or genotype representations. In the last decades, a plethora of EAs has been manually designed by domain experts for solving specific problems. Many noteworthy EA libraries have been made freely available and used to construct problem-specific EAs. For example, this includes the GAlib[4], EOlib[5] [69], and EASEA[6] [25]. The presence of EA libraries helps to reduce the amount of programming efforts and time required by providing commonly used genotype representations and evolutionary operators for rapid development of EAs. Although EA libraries can be used to construct different algorithms, they do not provide

[4] `http://lancet.mit.edu/ga`
[5] `http://eodev.sourceforege.net`
[6] `http://sourceforge.net/projects/easea`

the means to describe EAs on an abstract level, i.e., to create blue-prints that highlight the characteristic features of EAs. Domain knowledge used to design an EA is reflected by its implementation (i.e., source code) and thus inseparably embedded into the EA.

A different approach would be to separate domain knowledge from the implementation of the EA. Given i) a high-level description language to specify relevant problem-specific meta information and ii) an appropriate algorithm that is capable of interpreting the language, EAs, that are not *a priori* specialised for a particular problem but capable of exhibiting specialised behaviour based on the meta information provided about the problem at hand, can be constructed. An example for high-level approaches are grammar-based approaches, such as grammar-based genetic programming [130] or grammatical evolution [106]. Different problems can be supported by providing problem-specific grammars. For example, Jung and Reggia present a descriptive encoding language that makes use of a problem-specific grammar which can be used to evolve designs of neural networks [65]. Their high-level descriptive language, however, is specially designed for neural networks and is thus not directly applicable to other domains. A more general approach is described by Veenhuis et al. which allows specification of EAs on an abstract level using XML [127, 128]. Although this includes specification of genotypes and various algorithmic aspects (e.g., operators), it has several shortcomings. For example, it does not support grammars and constraints.

Existing approaches either do not consider any form of high-level specification or have been designed for a particular problem or branch of evolutionary computing. Although existing approaches, most notably grammar-based approaches, address some of these issues, a unified approach remains an open issue. For example, grammar-based approaches have been mostly used for genetic programming and it is not clear how applicable they are to problems from other domains. Therefore, in this chapter, we propose a formal evolutionary computing modelling language (ECML), based on the unified modelling

language (UML) [105], that allows the design of problem-specific genotypes. This includes specification of the structure and encoding of genotypes as well as relevant details regarding the processing of genotypes. In addition to the proposed ECML, we introduce the concept of a meta evolutionary algorithms (MEAs) as a class of EAs that is capable of interpreting ECML-based genotype models and processing genotypes accordingly. MEAs are meta algorithms because their search mechanisms and behaviour are defined by the meta information specified in the genotype model. Furthermore, MEAs can be designed as interpreters. This allows them to dynamically adapt their behaviour as the meta information is changing.

3.2.3 Meta Information: Genotype Representations, Operators, and Constraints

An important issue for improving the performance of an EA is the selection of an appropriate genotype representation [115]. A good representation captures important features of the problem domain that are necessary to effectively solve the problem [31]. To date, there exists a large variety of genotype representations. For example, string and matrix structures are commonly found in the context of genetic algorithms, evolutionary programming, evolutionary strategies, and learning classifier systems. In genetic programming, on the other hand, tree structured genotypes are used. Encoding of genotypes includes binary-, integer-, real-, and mixed coding. More recently, some work has encoded information regarding the logical structure as part of the genotype. For example, analog genetic encoding [92], used to evolve analog circuits and networks (hence the name), incorporates special tags into the genotype to indicate the type of information that follows. Nevertheless, meta information regarding the structure and encoding of the genotype is usually not provided.

Most components of an EA require knowledge about the genotype and its representation. For example, this includes variation operators, i.e., recombination and mutation

operators. In addition to selecting an appropriate representation it is also important to select appropriate operators that suit the problem and the chosen genotype representation. Some variation operators are more generally applicable than others. For example, a uniform crossover operator can be applied regardless how genotypes are encoded. In contrast, arithmetic crossover operators are only useful for integer- or real-coding genotypes. There are also EA components that generally do not require any knowledge about the genotype. For example, selection operators are concerned with the quality of individuals (in terms of some fitness metric) and thus do not require knowledge about the genotype representation.

Another important issue that has to be considered is constraint handling. There are various methods that are commonly used for constraint handling. For example, this includes penalising or repairing genotypes that do not satisfy constraints. Another constraint handling method is to use an indirect encoding of solutions to the problem. Moreover, it is also possible to make use of knowledge about constraints to design operators that prevent the generation of genotypes that do not satisfy the constraints altogether. This method requires the EA to be aware of constraints. For example, in the context of high-level approaches, attribute grammars can be used to specify semantic information that is exploited by the EA when evolving the genotypes [24].

Domain knowledge that has been used at design time to construct an EA is reflected by the choice of genotype representation, operators, and constraint handling methods. As explained above, we aim at separating this kind of knowledge from the actual implementation. We believe that the genotype representation is essential in this regard as many components of an EA rely on knowledge about the genotype. For this reason, we have chosen a genotype-centric approach and focus on the specification of genotype models using ECML. This includes not only meta information regarding the structure and encoding of genotypes but also meta information regarding the various processing aspects and constraints.

3.3 Genotype Specification

We aim at proposing a formal modelling language which can be widely accepted within the community. The unified modelling language (UML) is a general-purpose modelling language and has been chosen as the basis for ECML because it is a formal language (all elements have a well defined meaning). It is an industry standard, extensible, and supports constraints. In addition, UML is neutral because it is not associated with a specific problem or branch of evolutionary computing. An introduction to UML, including references to corresponding literature, can be found in [42].

UML provides a formalism that allows the modelling of the various artefacts of a system. This formalism includes various standardised modelling elements and a well-defined notation. Although its original purpose is to model software systems, UML can also be used to model non-software systems. For example, the Systems Modelling Language (SysML) is a general-purpose language that uses a subset of UML elements to support the specification, analysis, design, and verification of complex systems [104]. Similarly, ECML is a general-purpose language that is used to model genotypes and related processing aspects of an EA. For this purpose, ECML uses basic UML elements and introduces extensions that are specially needed in the context of evolutionary computing.

Two important advantages of UML are the support of constraints and its extensibility by means of stereotypes and profiles:

- Constraints may be expressed as plain text or by formal means. For example, the object constraint language [103], which supplements UML, can be used for this purpose. Constraints are enclosed by curly brackets (i.e., {`constraint`}) and attached to the corresponding modelling element.

- Stereotypes are used to extend the core semantics of UML with semantics that are needed in the context of a particular domain. A stereotype is enclosed by guillemets

(i.e., `<<stereotype>>`) and is attached to the corresponding modelling element. By doing so, the modelling element is marked as one that has a specific meaning in the domain context.

- Profiles are specifications of common model elements within a specific domain context. For example, SysML is defined as a profile which extends UML by a set of elements that are specifically needed for modelling complex systems.

In the remainder of this section, we will describe various modelling elements that are specially needed in the context of evolutionary computing. We also introduce a number of semantic extensions using corresponding stereotypes.

3.3.1 Genotype Encoding and Structure

A genotype model defines the structure and encoding of genotypes in the context of a particular problem. Based on the structural information provided by a genotype model, a genotype instance $g \in \mathcal{G}$ can be generated, where $\mathcal{G}$ is the set of all genotype instances that can be generated based on a specific genotype model G. Later in Section 3.4.1 we explain how exactly genotype instances are generated based on a given genotype model. As for the specification of genotype models, ECML provides two kinds of modelling elements that can be used for this purpose: entities and relationships.

3.3.1.1 Entities

Entities are used to define the various building blocks, such as genes, that are needed to model genotypes for a particular problem.

A gene is defined as "a locatable region of genomic sequence, corresponding to a unit of inheritance, which is associated with regulatory regions, transcribed regions and/or other functional sequence regions" [109]. Alleles are the different possible forms of a gene. Phenotypic traits depend on how corresponding genes are expressed, i.e., which

alleles are present at the particular locations of the genome that are associated with the corresponding genes. For example, the eye colour in humans is a phenotypic trait and depends on how the genes, that are associated with encoding the eye colour, are expressed by corresponding alleles (e.g., blue or green). A genome is the complete genetic information of an individual.

Similarly, in the context of evolutionary computing, we consider genes as the basic building blocks that constitute the genotype of an individual. In addition, alleles are used for coding purposes. A genotype consists of a number of genes that, depending on their location in the genotype, are associated with different problem-specific features. They are identified by the `<<gene>>` stereotype. The notation for genes is illustrated in Figure 3.1a. For example, consider the family of knapsack problems [70]. This kind of problem is concerned with the optimal selection of items, each being associated with a profit and a weight, in order to maximise the overall profit while not exceeding a total weight limit. Genotypes for the knapsack problem can be modelled by using genes to encode the selection of items. Each gene is expressed by an allele that indicates whether a particular item is selected or not. The genotype is thus a sequence of "yes" and "no" alleles in which the i-th allele corresponds to the i-th item.

We distinguish between two types of alleles: symbolic alleles and numerical alleles. For example, in the knapsack problem described above, symbolic alleles are used to indicate selection. Unlike symbolic alleles, numerical alleles can be used to perform arithmetic operations. Valid numerical values for expressing a particular gene are modelled by an interval with lower bound lb, upper bound ub, and resolution res. The resolution is used to specify the accuracy of the numerical values (i.e., it determines the number of discrete values). An interval thus represents a set of numerical values $\{lb, lb + r, lb + 2r, \ldots, ub - 2r, ub - r, ub\}$. We further distinguish between integer- and real-value numerical alleles, identified by the `<<int(`lb`,`ub`,`res`)>>` and the

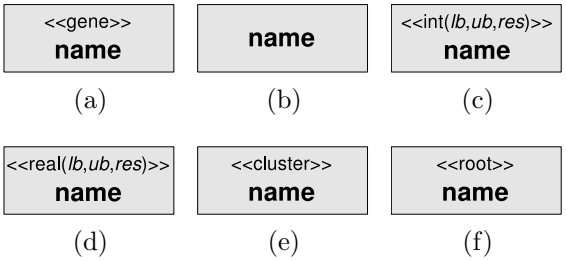


Figure 3.1: Notation of modelling elements used for genes (3.1a), symbolic alleles (3.1b), integer-value alleles (3.1c), real-value alleles (3.1d), cluster entities (3.1e), and root entities (3.1f).

`<<real(`lb`,`ub`,`res`)>>` stereotypes, respectively. Symbolic alleles are considered as default type and thus not explicitly marked as such. The notations for the various types of alleles are illustrated in Figure 3.1b through Figure 3.1d.

Multiple building blocks can be grouped together and treated as a higher-order building block. For this purpose, the concept of clusters is introduced. Unlike genes, cluster entities are not used for coding purposes, i.e., they are not expressed by alleles. Instead, they are used to represent higher-order building blocks. Cluster entities play an important role for structuring genotypes. For example, later in Section 3.5.2 we discuss a representation for a job-shop scheduling problem that makes use of cluster entities to structure the genotype. A cluster entity is identified by the `<<cluster>>` stereotype. A cluster with a special meaning is the root entity which indicates the starting point for interpreting the genotype model. It has to appear exactly once in a genotype model and is identified by the `<<root>>` stereotype. The notations for cluster and root entities are illustrated in Figure 3.1e and Figure 3.1f, respectively.

3.3.1.2 Relationships

ECML defines three kinds of relationships: expression, association, and activation relationships.

Expression Relationships When specifying the genotype model for a particular problem, it is necessary to indicate which alleles can be used to express a certain gene. For this purpose, the expression relationship is used. It connects an allele with a gene. Depending on the problem, there might be several alternative alleles for expressing a gene. For example, in the knapsack example above, a gene can be expressed by either one of two possible alleles. In such a case, there can be multiple expression relationships, involving the same gene and different alleles, to define the possible alternatives for expressing the gene. The notation for expression relationships is illustrated in Figure 3.2a. This example illustrates a gene E which can be expressed by a symbolic allele a.

Association and Activation Relationships Many problems require the definition of multiple genes of the same kind. For example, consider the knapsack problem described above. Depending on the number of selectable items, there has to be a corresponding number of genes in the genotype. For this purpose, association relationships are used. They establish one-to-many relationships between two building block entities (i.e., gene or cluster entity). This allows to specify the multiplicity of a building block relative to another. The exact multiplicity is an attribute of the relationship. The notation for association relationships is illustrated in Figure 3.2b. This example illustrates two genes E_1 and E_2 which are associated with each other. For each instance of gene E_1 there are exactly m instances of gene E_2. The naming convention for association relationships is R_{name}, where $name$ is the name of the relationship.

Another relationship, which is similar to an association relationship, is the activation relationship. It indicates whether a particular building block is present in the genotype or not, depending on the expression of a particular gene. Unlike an association relationship, which connects two gene or cluster entities with each other, an activation relationship connects an allele with a gene or cluster entity. Similarly to association relationships,

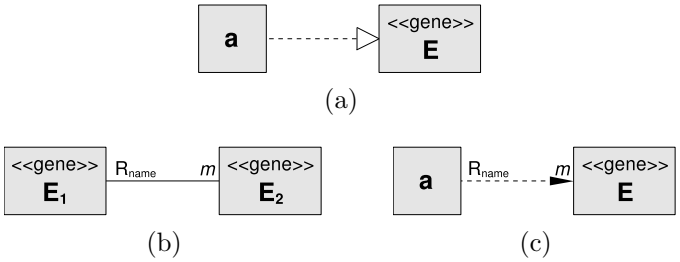


Figure 3.2: Notation of relationships used in the genotype model for expression (3.2a), association (3.2b), and activation (3.2c).

it is possible to specify the multiplicity of an activation relationship. The notation for activation relationships is illustrated in Figure 3.2c. This example illustrates an allele a which activates m instances of gene E. For each appearance of allele a there are m instances of gene E in the genotype. The naming convention for activation relationships is the same as for association relationships.

3.3.1.3 Example Genotype Models G_1 and G_2

The genotype model G_1, illustrated in Figure 3.3a, consists of two genes E_1 and E_2. Corresponding expression relationships indicate that gene E_1 can be expressed by the symbolic alleles a or b. Furthermore, it is indicated that gene E_2 can be expressed by the symbolic alleles c or d. There are three instances of gene E_1 in every genotype instance created, based on this model. This is indicated by the association relationship R_1 which connects the root entity G_1 with gene E_1. Whether or not there will be any instances of gene E_2 in the genotype instance depends on how the various instances of gene E_1 are expressed. For each instance of gene E_1 which is expressed by symbolic allele b, there are two more instances of gene E_2. This is indicated by the activation relationship R_2.

Although the size of genotype instances for genotype model G_1 is not fixed, it is limited to a maximum of three E_1 and six E_2 instances. In contrast, the genotype model G_2, illustrated in Figure 3.3b, can grow arbitrarily large during the process of evolution. Genotype instances based on genotype model G_2 have at least three E_1 instances (because

of association relationship R_1). For each E_1 instance there is a E_2 instance (because of association relationship R_2) and depending on how this gene is expressed (either y or n), there are more E_1 instances (because of activation relationship R_3). Circular relationships, such as in genotype model G_2, effectively allow the genotype structure to grow during the process of evolution.

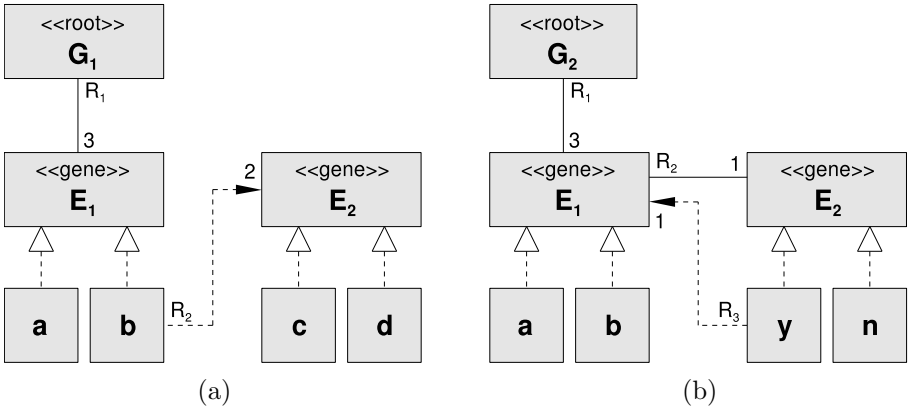


Figure 3.3: Example for genotype models using non-circular relationships (3.3a) and circular relationships (3.3b).

The actual genotype representation depends on the EA implementation. Later in Section 3.4.1 we discuss our proof-of-concept MEA implementation and explain how these two examples are represented.

3.3.2 Genotype Processing Meta Information

The various entities and relationships described above can be used to model the problem-specific genotype structure and its encoding. In addition, a number of extensions are introduced in order to specify meta information regarding the various processing aspects of an EA.

3.3.2.1 Variation Operators

An important aspect of an EA is the creation of new individuals. This is done by applying variation operators (i.e., recombination and mutation) to the selected indi-

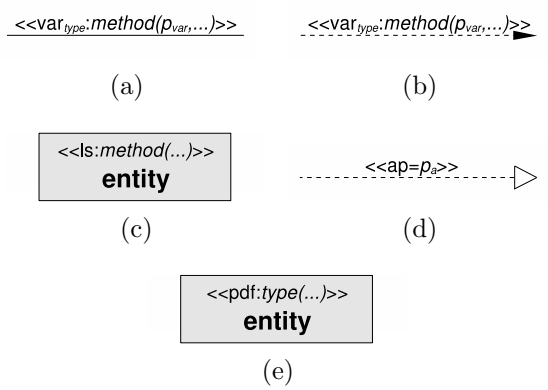


Figure 3.4: Notation of extensions for specification of meta information regarding processing aspects: "var" stereotypes assigned to an association relationship (3.4a) and an activation relationship (3.4b), a "ls" stereotype assigned to an entity (3.4c), an "ap" stereotype assigned to an expression relationship (3.4d), and a "pdf" stereotype assigned to an entity (3.4e).

viduals of the population. What kind of variation operations to use depends on the problem. ECML allows to assign variation operators to association and activation relationships by using a corresponding stereotype. The "var" stereotype is defined as $\texttt{<<var}_{type}\texttt{:}method\texttt{(}p_{var}, v_0, \ldots, v_{n-1}\texttt{)>>}$, where *method* indicates the particular kind of variation operator, p_{var} is the probability that the method is performed, and $v_0, \ldots, v_{n-1}$ are optional parameters that might be required by the method. With *type* we further distinguish between "var" stereotypes for recombination $\texttt{<<var}_{\texttt{rc}}\texttt{:...>>}$ and mutation $\texttt{<<var}_{\texttt{mt}}\texttt{:...>>}$.

Multiple variation operators might be applicable to the same relationship. For this purpose, it is possible to attach multiple "var" stereotypes to the same relationship. In addition, it is possible to specify different recombination operators for different association or activation relationship. Effectively this means that different variation operations are performed for different parts of the genotype. This applies to recombination operators and mutation operators likewise. If no variation operator is specified along a particular relationship, then there will be no such operation performed for the corresponding part

of the genotype. The notation of the "var" stereotype, attached to an association relationship and an activation relationship, is illustrated in Figure 3.4a and Figure 3.4b, respectively.

3.3.2.2 Local Search

EAs are sometimes used in combination with a local search method in order to improve the performance. These so-called memetic algorithms [107] have become popular because their performance is often significantly better as compared to EAs that do not perform local search. Depending on the problem, different kinds of local search methods have to be used. For this purpose, we introduce the "ls" stereotype which indicates that a specific local search method is applied. This stereotype is defined as `<<ls:`$method(p_{ls}, v_0, \ldots, v_{n-1})$`>>`, where *method* indicates the particular kind of local search method, p_{ls} is the probability that the method is performed, and $v_0, \ldots, v_{n-1}$ are optional parameters that might be required by the method. The notation of the "ls" stereotype is illustrated in Figure 3.4c.

3.3.2.3 Allele Probability Distribution

When creating or mutating genotypes, alleles are often chosen randomly according to some probability distribution. Usually these probabilities are uniformly distributed among all possible alleles. It is possible to specify a non-uniform allele probability distribution by using domain knowledge. For this purpose, we introduce the "ap" stereotype which is attached to any expression relationship and allows the specification of a non-uniform allele probability distributions. It is defined as `<<ap=`p_a`>>`, where p_a is the probability of the specific allele to be selected. As a result, some alleles are selected more frequently than others. This effectively causes the search to be biased towards a certain region in the search space. The notation of the "ap" stereotype is illustrated in Figure 3.4d.

In case there are multiple numerical alleles for a gene, then one allele is selected with a corresponding probability p_a. Effectively, this means that a particular interval is selected with a probability p_a. However, for the selected interval, all possible values are equally likely. This means that a uniform probability distribution function is implicitly assumed for selecting a specific value within a particular value range. Alternatively, it is possible to specify a customised probability distribution function using the "pdf" stereotype. It is defined as `<<pdf:`*type*$(v_0, \ldots, v_{n-1})$`>>`, where *type* is the particular distribution function to be used and $v_0, \ldots, v_{n-1}$ are optional parameters that might be required by the probability distribution function. The notation of the "pdf" stereotype is illustrated in Figure 3.4e. Note that the use of the "pdf" stereotype is only meaningful for numerical alleles.

3.4 A Meta Evolutionary Algorithm: MEAα

MEAs refers to a family of EAs that is capable of interpreting high-level specifications (i.e., genotype models presented in ECML) and behaving accordingly. The behaviour of a MEA can thus change during runtime if the information provided by the genotype model changes. This includes changes regarding operators (e.g., recombination, mutation, and local search) as well as changes in the genotype structure or its encoding. Different problems require different operators. Therefore, a MEA has to be equipped with a repository of supported operators. For example, EA libraries such as the ones mentioned in Section 3.2.2 can be used for this purpose. During runtime, corresponding operators are selected on demand. Selection of operators is based on the meta information provided by the genotype model, which specifies what operators can be applied to a particular part of the genotype.

We have developed MEAα, a proof-of-concept Java implementation of MEA, which is used as implementation of f_{opt}, further denoted as $f_{opt}^{MEA\alpha} : \mathcal{G} \rightarrow \mathcal{G}_o$. In other words,

Table 3.1: Overview of the abstract concepts and their corresponding concrete counterparts

Abstract Concept	Concrete Counterpart
f_{opt}	$f_{opt}^{MEA\alpha}$
Γ	G
x_i	g_i
X	$\mathcal{G}$
$X_o \in X$	$\mathcal{G}_o \in \mathcal{G}$

MEAα is used as the optimisation method needed to perform the WIA process (see Section 2.5). When using our approach, based on ECML and MEAα, a candidate solution x_i is encoded by a corresponding genotype instances g_i. For the remainder of this dissertation we distinguish between the terms "solution" and "genotype instance" as follows. When discussing general and theoretical issues concerning problem solving in symbiotic simulation on an abstract level, we use the term "solution", denoted by x. A genotype is a specific way of encoding solutions in the context of the evolutionary computing approach, used in this dissertation. When discussing implementation-specific issues concerning our approach we will thus use the term "genotype instance". Table 3.1 presents an overview of the abstract concepts discussed in Chapter 2 and their corresponding concrete counterparts introduced in this chapter.

MEAα is designed to be an interpreter that adapts its behaviour dynamically depending on the meta information provided by the genotype model. While the schematic outline of MEAα follows that of standard EAs, its exact behaviour depends entirely on the information specified in the genotype model. For examples, whether or not a particular local search operator is applied depends on whether a corresponding "ls" stereotype is specified in the genotype model. Similarly, what kind of variation operation is applied to a particular part of the genotype depends on the corresponding "var" stereotype associated with a particular relationship. Some aspects of an EA are not covered by ECML in its current form. For example, parent selection and termination conditions are not

specified by ECML. For our proof-of-concept implementation MEAα we make use of a tournament operator to perform parent selection. In addition, a fixed number of generations is used as termination condition. An outline of MEAα is illustrated in Figure 3.5.

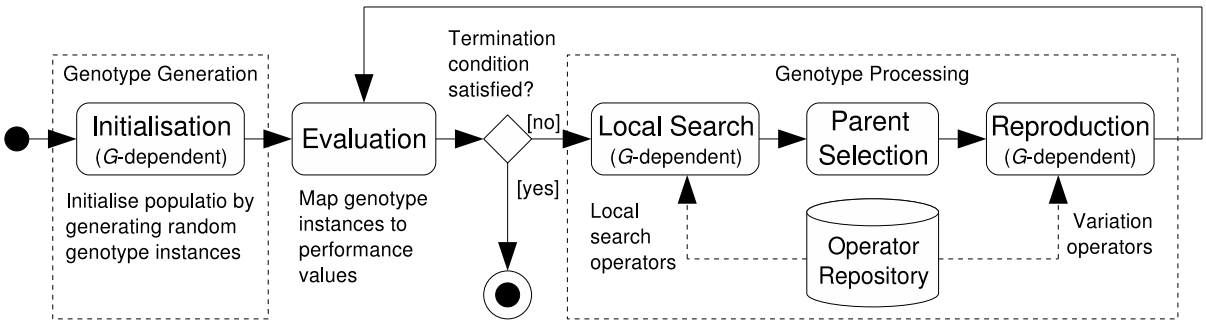


Figure 3.5: Outline of MEAα. Corresponding local search and variation operators are provided by a repository and chosen during runtime. Initialisation, local search, and reproduction make use of information provided by a genotype model G.

3.4.1 Genotype Generation

The genotype instances generated by MEAα are hierarchically structured as a direct result of the one-to-many relationships in the genotype model, i.e., genotype instances are tree structures. The various entities and relationships in the genotype model are reflected by nodes and links in genotype instances, respectively.

The initial population is created by generating a number of random genotype instances. A random genotype instance is created by parsing the genotype model and creating a corresponding number of nodes for each entity that is encountered. The exact number of nodes that are created depends on the multiplicity m of the corresponding association or activation relationship. Each of the created nodes are child nodes that are connected to their parent node by a corresponding link set. Parent and child nodes are nodes that correspond to the entities indicated on the source (from entity) and the target

(to entity) of a one-to-many association/activation relationship, respectively. Each child node will be assigned a node index $0 \ldots m-1$ which allows to identify nodes that are connected by the same link set.

Parsing of the genotype model is done in a depth-first fashion by following outgoing association and activation relationships of all entities in the genotype model, starting with the root entity. Once an entity instance is created, all outgoing relationships and connected entities are processed in a recursive fashion. If a gene is encountered, then a random allele is selected from the set of possible alleles (determined by considering the various expression relationships for that particular gene). A uniform allele probability distribution is assumed unless specified otherwise by using the "ap" stereotype. In case of a numerical allele, a random value is generated uniformly with respect to the given interval unless specified otherwise by using the "pdf" stereotype.

For presentation purposes, we use a tree representation of genotype instances. Nodes that reflect gene entities are represented by rounded rectangles and named by the allele and the name of the gene, separated by a colon (i.e., *allele_name*:*gene_name*). For numerical alleles the actual value of the allele is included as well (i.e., *allele_name* = *value*:*gene_name*). Nodes that reflect cluster entities are represented by circles named after the cluster. An exception are nodes reflecting root entities which are named by the name of the genotype instance. Each node (except root nodes) has a node index assigned to it. In our notation of nodes, we indicate the node index for nodes that reflect genes or clusters in the upper right corner or at the top, respectively. A set of links connects a node with its child nodes. These link sets are named according to the relationship they reflect: $L_{name,i}$, where *name* is the name of the corresponding relationship in the genotype model and i the index of the link set in case there are multiple link sets that correspond to the same relationship.

For example, consider the genotype model G_1 illustrated in Figure 3.3a. A random genotype instance is created by starting with the root entity G_1. A corresponding root

node which reflects the name of the genotype instance is created. Association relationship R_1 connects root entity G_1 with gene entity E_1. In addition, R_1 has a multiplicity of 3. Therefore, three nodes are created by selecting random alleles (i.e., either a or b). These nodes are connected to the root node by a set of links $L_{1,0}$. Entity E_1 has no outgoing association relationships. If the selected allele does not have any outgoing activation relationships either, then no further processing has to be done. However, in case of a b allele, there is one outgoing activation relationship (i.e., R_2). This relationship is processed in the same way as just described for relationship R_1. This process continues until the genotype of the individual is complete, i.e., if all entities and relationships in the genotype model have been processed. Figure 3.6 illustrates examples of valid genotype instances for the genotype models G_1 and G_2, illustrated in Figure 3.3.

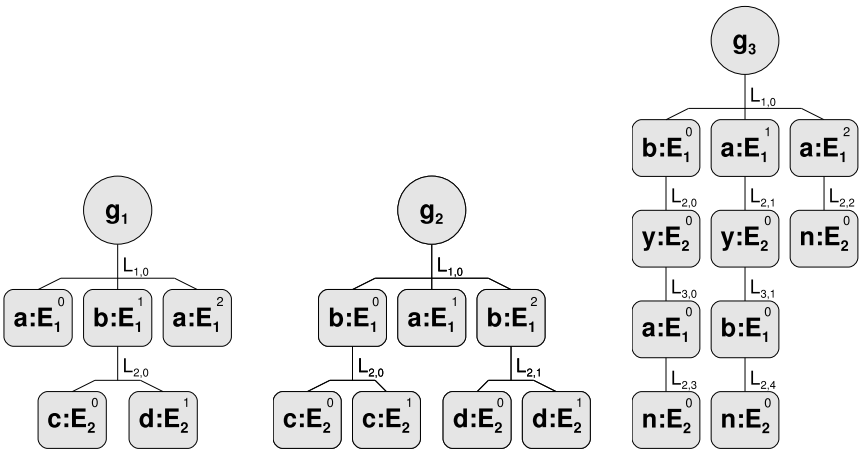


Figure 3.6: Examples of possible genotype instances g_1 and g_2 for genotype model G_1 and a possible genotype instance g_3 for genotype model G_2.

3.4.2 Genotype Processing

Genotype processing in MEAα is concerned with performing local search and reproduction by means of corresponding operators. The meta information provided by the genotype model determines which operator is used for processing genotype instances.

More precisely, the various "ls" and "var" stereotypes are used for this purpose. Operators are selected with a certain probability, specified in the corresponding stereotype (i.e., p_{ls} and p_{var}). As explained above, different operators can be applied to different parts of the genotype. In general, MEAα performs recombination by sub-tree swapping. The multiplicity of an association or activation relationship indicates the number of corresponding child nodes in the genotype instances. If recombination is performed, then these child nodes are representing the root nodes of the sub-tree structures which are subject for swapping. The exact way in which sub-tree structures are swapped between two parent genotype instance depends on the recombination operator.

For example, consider genotype model G_1 and the two possible genotype instances g_1 and g_2. Assume that a one-point crossover operation can be performed along relationship R_1. Since this relationship has a multiplicity of $m = 3$, there are three child nodes, each reflecting an instance of a E_1 gene, in the genotype instance. These child nodes represent the root nodes of the sub-tree structures used for swapping. Since a one-point crossover operation is performed, a crossover point $0 < k < m$ is randomly chosen. This means that all sub-tree structures with root nodes that have an index $j \geq k$ will be swapped between the two parent genotype instances g_1 and g_2. Figure 3.7 illustrates how MEAα performs recombination for this example.

Mutation is performed in a similar fashion as recombination except that genotype instances are processed by completely replacing randomly selected sub-trees rather than by swapping them. For this purpose, randomly selected nodes (and all their child nodes) are removed from the genotype instance and replaced by new nodes. This node and all its child nodes (if any) are generated in the same fashion as discussed above in the context of genotype generation.

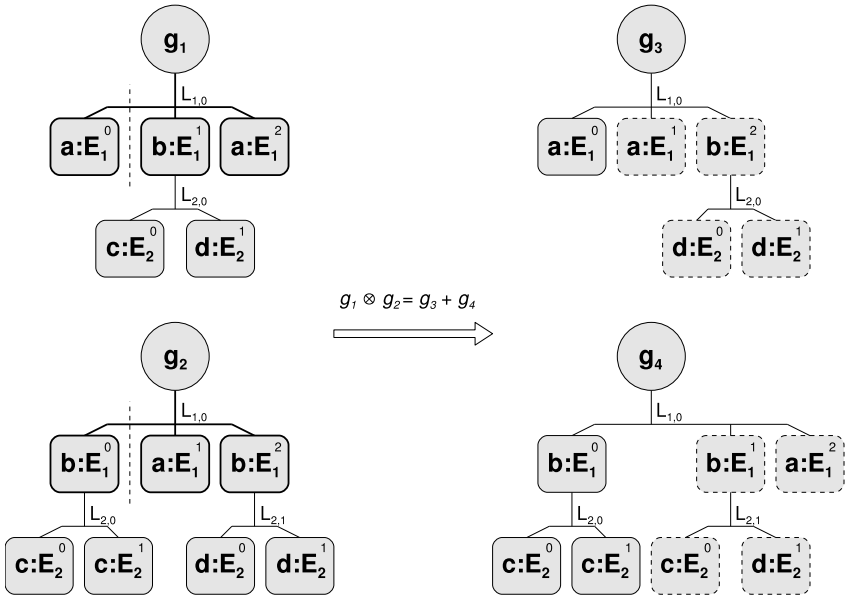


Figure 3.7: Example for recombination of genotype instances g_1 and g_2 by performing sub-tree swapping. In this example, it is assumed that a one-point crossover operation is performed along relationship R_1. The corresponding link sets $L_{1,0}$ for relationship R_1 and the root nodes of the sub-tree structures are highlighted. A crossover point $k = 1$ is indicated by a dashed line that separates nodes with index $j \geq k$ from the node with index $j = 0$. In addition, the sub-tree structures that have been swapped during recombination are indicated by dashed lines in the resulting genotype instance g_3 and g_4.

3.5 Application Examples

One of the objectives of this research is to provide a symbiotic simulation framework that is applicable to a wide variety of applications and not limited to a specific type of application. This requires our approach to be applicable to a wide range of different classes of problems. MEAα can be applied to problems across different domains and branches of evolutionary computing. We demonstrate this by applying our approach to the following problems:

- *Knapsack Problem.* A well-known standard optimisation problem which is of interest due to its applications in industry and finance management. In this example we show how our approach can be used to incorporate problem-specific knowledge

by using different allele probabilities.

- *Job-Shop Scheduling Problem.* Another well-known standard optimisation problem which is of interest due to its applications in scheduling and manufacturing. In this example, we show how different representations of the same problem can be described in ECML. In addition, we show how cluster entities can be used to structure the genotype and how ECML can be used to specify constraints.

Fully-fledged application examples that make use of simulation will be discussed in Chapter 5 and Chapter 6. In this section we consider problems that do not require simulation for evaluation. Therefore, we can simplify the evaluation process and consider a problem-specific evaluation function $f_e : g_i \rightarrow PI_i$, which maps a genotype instance to a corresponding PI value without the need to compose a what-if scenario and perform a simulation. For each of the problems discussed in this section, we will explain how the evaluation function f_e is implemented. Furthermore, we will also explain how the comparison function f_c compares the various PI values used to rank genotype instances.

3.5.1 Multi-Dimensional Knapsack Problem

Knapsack problems are concerned with finding the optimal combination of items for which the profit is maximised while the total weight must not exceed the capacity of the knapsack(s). This kind of problem has been of interest because of its applications in industry and finance management. There are many variations of the knapsack problem such as the 0-1 knapsack problem, the bounded knapsack problem, and the unbounded knapsack problem, for instance. Depending on the particular kind of knapsack problem there can be various constraints which make the problem more difficult to solve. The particular kind of knapsack problem with which we will be concerned here is the multi-dimensional knapsack problem (MKP). Multiple knapsacks with different capacities are

considered. If an item is selected, then it contributes to the total weight of every knapsack, where it may take up more or less space depending on the knapsack. Formally, the MKP can be stated as:

$$\text{maximise} \quad \sum_{j=1}^{m} p_j a_j, \tag{3.1}$$

$$\text{subject to} \quad \sum_{j=1}^{m} r_{ij} a_j \leq b_i, i = 1, \ldots, n, \tag{3.2}$$

$$\text{and} \quad a_j \in \{0, 1\}, j = 1, \ldots, m, \tag{3.3}$$

where m and n are the number of items and knapsacks, respectively. If selected (i.e., if $a_j = 1$), then the j-th item contributes to the total profit with its individual profit p_j and is placed in all knapsacks where it takes up a certain amount r_{ij} of space. An arbitrary number of items can be selected as long as none of the knapsack capacities b_i is exceeded. The objective is to select a subset of items for which the total profit is maximised. The profits p_j and the space requirements r_{ij} are dependent on the specific problem instance under consideration. Later in Section 3.5.1.3, we apply our approach to a total 48 well-known problem instances from the literature.

3.5.1.1 Representation

The genotype structure of the knapsack problem is rather simple. Items are represented by corresponding E_{item} genes, each of which can be expressed by either a *yes* or *no* allele. Symbolic alleles have been chosen because selecting items is a discrete decision which does not require numerical encoding. Each of the possible outcomes of this decision is represented by a unique symbolic allele. The choice of symbols is irrelevant as long as the decoding mechanism is able to distinguish between them. A single association relationship R_{items} is used to connect the root element with the E_{item} gene. This relationship allows us to specify the multiplicity m, i.e., the number of items considered for a particular problem instance. The general genotype model $G_{KS,G}$ for the knapsack problem is illustrated in Figure 3.8.

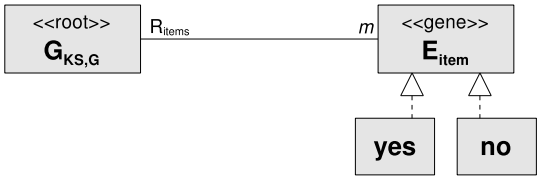


Figure 3.8: General genotype model $G_{KS,G}$ of the knapsack problem, defining the structure and encoding of the genotype.

In addition to specifying the structure and encoding of genotypes, it is necessary to add meta information regarding the processing aspects. For example, we need to specify information regarding the various variation operators. In case of the knapsack problem, a standard recombination operator such as the one-point crossover, indicated by `<<var_rc:onepoint(0.6)>>`, works well. The probability that this operator is applied when processing genotypes is $p_{var} = 0.6$. The operator does not require any additional parameters. Similarly, we specify that a random mutation operator is applied with a probability of $p_{var} = 1.0$ using the `<<var_mt:rand(1.0, 1/m)>>` stereotype. This mutation operator has an additional parameter $v_0 = 1/m$ that indicates the mutation rate (i.e., the probability for mutating alleles) used by the operator.

Domain knowledge can be used to improve the performance for solving particular problem instances. For this purpose, it is sometimes useful to bias the selection of alleles during generation of individuals or performing mutation. For example, Khuri et al. bias their genetic algorithm to a particular problem instance by specifying a probability of 5% that an item is selected [74]. By doing so, they are able to achieve better results as compared to an unbiased version of their genetic algorithm. Such a bias can be specified in the genotype model by using the "ap" stereotype. The specialised genotype model $G_{KS,S}$, is illustrated in Figure 3.9, reflects this improvement.

3.5.1.2 Evaluation Function f_e and Comparison Function f_c

As for the decoding of a genotype instance, $a_j = 1$ if the j-th E_{item} gene is expressed by a *yes* allele. Evaluation of a genotype instance g is then done by considering the total

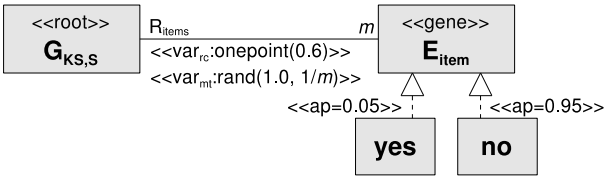


Figure 3.9: Specialised genotype model $G_{KS,S}$ of the knapsack problem featuring stereotypes that indicate the choice of variation operators and a customised allele probability distribution.

profit $P(g)$, the feasibility $F(g)$, and the remaining unused capacity $C(g)$:

$$P(g) = \sum_{j=1}^{n} p_j a_j \tag{3.4}$$

$$F(g) = \begin{cases} \texttt{true} & \text{if } \forall_{i=1\ldots m} : \sum_{j=1}^{n} r_{ij} a_j \leq b_i \\ \texttt{false} & \text{otherwise} \end{cases} \tag{3.5}$$

$$C(g) = \sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} r_{ij} a_j \right) \tag{3.6}$$

The result PI for the evaluation of genotype instance g is:

$$PI = f_e(g) = \{P(g), F(g), C(g)\} \tag{3.7}$$

Two genotype instances g_a and g_b are compared based on their profit, feasibility, and remaining unused capacity. In general, any solution is feasible as long as the capacity constraints of none of the knapsacks is violated. It is possible that genotype instances are created, either by recombination or mutation, which are valid (i.e., they are compliant to the genotype model) but produce solutions that are not feasible. This is a common issue when solving knapsack problems with genetic algorithms and can be treated in various ways. For example, Khuri et al. allow genotype instances that represent infeasible solution to join the population and use a penalty term to decrease their fitness as compared to genotypes that represent feasible solutions [74]. In addition, infeasible genotype instances which are farther away from feasibility are penalised more heavily. We have chosen the

same approach to handle constraint (3.2), i.e., we penalise genotype instances of infeasible solutions as described below.

Feasible solutions are always considered better than infeasible ones:

$$F(g_a) = \texttt{true} \wedge F(g_b) = \texttt{false} \Rightarrow g_a \preceq g_b \tag{3.8}$$

If both solutions are feasible, i.e., if $F(g_a) = F(g_b) = \texttt{true}$, then the profit is used as comparison criterion:

$$P(g_a) \geq P(g_b) \Rightarrow g_a \preceq g_b \tag{3.9}$$

In case two feasible solutions have the same profit, i.e., if $F(g_a) = F(g_b) = \texttt{true} \wedge P(g_a) = P(g_b)$, or if both solutions are infeasible, i.e., if $F(g_a) = F(g_b) = \texttt{false}$, then the unused capacity is used as comparison criterion:

$$C(g_a) \geq C(g_b) \Rightarrow g_a \preceq g_b \tag{3.10}$$

3.5.1.3 Experimental Evaluation

MEAα is evaluated on 48 well-known problem instance from the literature. More specifically, we use the test problems introduced by Senju and Toyoda [122] ("sento1" and "sento2"), Weingartner and Ness [129] ("weing1" – "weing7"), Shih [124] ("weish01" – "weish30"), and Freville and Plateau [43] ("pb1" – "pb7", "hp1", and "hp2"). For all these problems, optimal (i.e., maximum) profit values are known from the literature and will be used as reference values for evaluation purposes. We determine the effectiveness ϕ^{MKP} of MEAα in finding the optimal solution:

$$\phi^{MKP} = \frac{P(g_o)}{P_o}, \tag{3.11}$$

where g_o is the best solution found by MEAα and P_o is the known maximum profit.

We use the general genotype model (illustrated in Figure 3.8). As for the parameters, we have chosen the same values as reported by Khuri et al. [74]. We use a population

size of 50 genotype instances and limit the evolutionary process to a maximum of 2000 generations. For each problem, a set of 100 experiments has been performed. The results obtained from these experiments are illustrated in Table 3.2 which shows the minimum and maximum profits (P_{min} and P_{max}) obtained (over the set of all experiments), as well as the corresponding minimum and maximum effectiveness (ϕ_{min}^{MKP} and ϕ_{max}^{MKP}). Some solutions are found more often than others.

Khuri et al. [74] have analysed a number of MKP instances, including the "weing8" problem instance. For this particular problem instance, they used a biased initial population so that the probability to generate a '0' bit when generating a random genotype is 0.95. In their case, a zero bit indicates that a particular item is not selected. This form of *a priori* knowledge can be easily integrated into an ECML model by using corresponding "ap" stereotypes. The specialised genotype model for this particular case is illustrated in Figure 3.9. We have performed two sets of experiments for this problem instance using the general and the specialised MKP genotype model. The result obtained using the specialised genotype model is indicated as "weing8b". Table 3.3 shows a performance comparison for solving the "weing8" problem instance when using the general genotype model and the specialised genotype model.

The P_{max} result obtained for "weing8", using the general genotype model, is 583775 which is 6.5% less as compared to Khuri et al. [74], who reported 624319 for this problem instance. However, when using the specialised genotype model, the obtained results are improved to $P_{max} = 623612$ for "weing8b". This example highlights the importance of problem-specific knowledge in order to improve the evolutionary process. With ECML it is possible to specify problem-specific knowledge in various ways. In this example we have used a problem-specific allele probability distribution.

Table 3.2: Effectiveness of MEAα in solving various standard MKP instances.

Problem	P_{min}	P_{max}	ϕ_{min}^{MKP}	ϕ_{max}^{MKP}
hp1	3258	3418	0.9532	1.0000
hp2	2923	3186	0.9175	1.0000
pb1	2862	3090	0.9262	1.0000
pb2	2931	3186	0.9200	1.0000
pb4	90262	95168	0.9484	1.0000
pb5	2051	2139	0.9589	1.0000
pb6	530	776	0.6830	1.0000
pb7	907	1035	0.8763	1.0000
sentO1	6476	7761	0.8332	0.9986
sentO2	8361	8720	0.9586	0.9998
weing1	132538	141278	0.9381	1.0000
weing2	118920	130883	0.9086	1.0000
weing3	83320	95627	0.8708	0.9995
weing4	103729	119337	0.8692	1.0000
weing5	92568	98796	0.9370	1.0000
weing6	117418	130623	0.8989	1.0000
weing7	1061135	1094767	0.9687	0.9994
weing8	523500	604347	0.8385	0.9680
weish01	4400	4554	0.9662	1.0000
weish02	4404	4536	0.9709	1.0000
weish03	4001	4115	0.9723	1.0000
weish04	4505	4561	0.9877	1.0000
weish05	4451	4514	0.9860	1.0000
weish06	5383	5557	0.9687	1.0000
weish07	5359	5567	0.9626	1.0000
weish08	5440	5605	0.9706	1.0000
weish09	4639	5246	0.8843	1.0000
weish10	6101	6339	0.9625	1.0000
weish11	5421	5643	0.9607	1.0000
weish12	6086	6339	0.9601	1.0000
weish13	5915	6159	0.9604	1.0000
weish14	6607	6954	0.9501	1.0000
weish15	6856	7486	0.9158	1.0000
weish16	7103	7288	0.9745	0.9999
weish17	8528	8633	0.9878	1.0000
weish18	9333	9580	0.9742	1.0000
weish19	7175	7667	0.9321	0.9960
weish20	9103	9450	0.9633	1.0000
weish21	8460	9074	0.9323	1.0000
weish22	8377	8897	0.9363	0.9944
weish23	7869	8322	0.9431	0.9974
weish24	9840	10220	0.9628	1.0000
weish25	9451	9939	0.9509	1.0000
weish26	8955	9552	0.9344	0.9967
weish27	9114	9819	0.9282	1.0000
weish28	8935	9469	0.9413	0.9976
weish29	8704	9331	0.9250	0.9916
weish30	10850	11187	0.9695	0.9996

Table 3.3: Comparison of "weing8" and "weing8b" using the general genotype model and the specialised genotype model, respectively.

Problem	P_{min}	P_{max}	ϕ_{min}^{MKP}	ϕ_{max}^{MKP}
weing8	523500	604347	0.8385	0.9680
weing8b	553130	623612	0.8860	0.9989

3.5.2 Job-Shop Scheduling Problem

The job-shop scheduling problem (JSP) is a well known problem, where n jobs have to be scheduled on m machines. A job $J_i = (O_{i,0}, O_{i,1}, \ldots, O_{i,m-1})$ has to be processed by each machine, i.e., there is a total of m operations which have to be performed on a job where $O_{i,k}$ represents the machine index of the k-th operation of the i-th job. The order in which jobs have to be scheduled on machines is different for each job. For example, consider the following example where $n = 2$ and $m = 4$: $J_0 = (0, 2, 3, 1)$ and $J_1 = (3, 1, 2, 0)$. Both jobs have to be processed by all four machines. However, the order in which they visit each machine is different. In addition, the k-th operation of the i-th job has a processing time $TP_{i,k}$. The problem is to find an optimal schedule for which the makespan, i.e., the total time to process all jobs, is minimised. Two constraints have to be considered. First, operations have to be performed in a specific sequence, i.e., a job has to be scheduled on all machines in a specific order. This is also referred to as the precedence constraint which states that a particular operation can only be performed if all preceding operations have already been performed. Second, a machine can only process one job at a time. Formally, the JSP can be stated as:

$$\text{minimise} \quad C(g) = TF_{i,m-1}, \quad \nexists j : TF_{j,m-1} > TF_{i,m-1} \tag{3.12}$$

$$\text{subject to} \quad \forall i, k : TF_{i,k-1} < TS_{i,k} \tag{3.13}$$

$$\text{and} \quad \forall 0 \leq l < m : max(a_l(t)) = 1, \quad t \geq 0 \tag{3.14}$$

where $C(g)$ is the makespan of a schedule obtained from a genotype instance g and $TS_{i,k}$ and $TF_{i,k}$ are the starting and finishing times of the k-th operation of the i-th job, i.e., $TF_{i,k} = TS_{i,k} + TP_{i,k}$. In addition, $a_l(t)$ indicates the number of jobs that are processed by the the l-th machine at a given time t. The first constraint is expressed in (3.13) which asserts that the starting time $TS_{i,k}$ of the k-th operation of the i-th job is after the finishing time of the previous operation of the same job. The second constraint is expressed in (3.14) which asserts that for any machine l, there is at most one job processed at any given time $t \geq 0$.

There are various possible representations for the JSP. In general, there are two approaches to model the problem. It is distinguished between a direct representation and an indirect representation, depending on whether the schedule itself is represented by a solution [49]. If the schedule is not directly represented in the genotype, then it is necessary to decode the genotype in order to obtain the actual schedule. For example, an indirect representation may only specify the order in which jobs are being processed on a particular machine. However, the exact starting and finishing times of jobs, i.e., the schedule, is not directly specified. Indirect representations are sometimes preferred over direct representations because of their simplicity. Here, we describe and compare two possible indirect representations for the JSP. With this example, we demonstrate how ECML can be used to model the same problem in different ways. This is an important feature of our approach as it gives the user the freedom to select whichever representation seems to be most appropriate, instead of dictating a specific representation.

3.5.2.1 Representation 1

The first JSP representation and its corresponding decoding scheme follow the one discussed in [48]. A rank, ranging from $1 \ldots m$, is assigned to the m operations of each of the n jobs. This rank is unique among the operations of a single job, i.e., there can

be only one operation within a job which is assigned a specific rank. Operations are decoded into schedules based on their ranks. For each machine, lower ranking operations are preferred over higher ranking operations. In addition, operations are sorted according to their processing times (shortest first) and, in case of conflicts, priority is given to jobs that arrived first. An operation is scheduled so that none of the constraints are violated. Depending on how ranks are assigned to operations, the resulting schedule is different. The problem is thus to find the optimal assignment of ranks to operations so that the makespan C is minimised.

The genotype model is partitioned into n job clusters. Each job in the genotype is associated with a sequence of m ranks, i.e., for each job there are m rank alleles. Each rank allele is associated with a particular operation. For example, the k-th allele in the sequence encodes the rank of the k-th operation. Within the context of a job, each rank must appear exactly once, i.e., a particular allele must appear only once within a job cluster. For this purpose, we introduce the "limited" constraint {`limited(`lb`,`ub`)`}. This constraint indicates how often a particular allele may appear within the scope of a cluster. For example, in the context of this JSP genotype representation, a {`limited(1,1)`} constraint is assigned to the various rank alleles, indicating that a particular rank allele must appear exactly once within the scope of a job cluster. However, since there are n job clusters, the same rank appears exactly n times within the genotype. Figure 3.10 illustrates the genotype model G_{JSP1} and example genotype instance g_5 of the first JSP representation.

In addition to structural information, domain knowledge is reflected by the different kinds of crossover operators used for the two relationships in the genotype model. The order in which the operations of a particular job are ranked is important. Therefore, a corresponding order crossover [28] is used: `<<var`$_{\texttt{rc}}$`:order(`p_{var}`)>>`. While the order of alleles within job clusters is important, this is not the case for job clusters. Therefore, a common uniform crossover operation can be applied: `<<var`$_{\texttt{rc}}$`:uniform(`p_{var}`,`

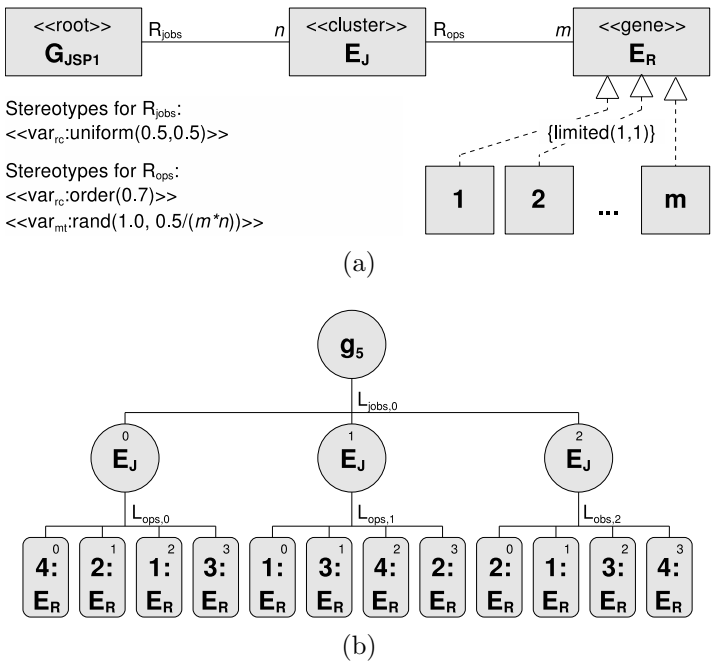


Figure 3.10: Genotype model G_{JSP1} (3.10a) and a possible genotype instance g_5 for $n = 3$ jobs and $m = 4$ machines (3.10b).

v_0)>>, where parameter v_0 is the probability that a crossover operation is performed for a particular gene E_J. The order and uniform crossover operations are associated with relationships R_{ops} and R_{jobs}, respectively. In addition, a mutation operator is associated with relationship R_{ops}: <<var$_{\texttt{mt}}$:rand(p_{var}, v_0)>>, where v_0 is the probability that a particular gene is randomly mutated. The probabilities for the various operators indicated in the genotype model have been chosen based on experimental experience.

3.5.2.2 Representation 2

The second JSP representation has originally been introduced by Bierwirth [10]. In this representation, a solution is encoded as an unpartitioned string in which each gene represents a job. Each job appears exactly m times which is ensured by using a {limited(m,m)} constraint. By scanning a genotype instance from left to right, the k-th appearance of job j refers to the k-th operation of this job. The order in which operations are performed is important. For this purpose, the same order crossover operation,

as used previously in Section 3.5.2.1, is used along relationship R_{ops}. The probabilities for the various operators indicated in the genotype model have been chosen based on experimental experience. Like the previous JSP representation, this representation is an indirect representation and has to be decoded into a schedule. The advantage of this representation is that it never violates the precedence constraint because alleles represent job indices. Therefore, the k-th occurrence of allele i, refers (by definition) to the k-th operation of the i-th job. Figure 3.11 illustrates the corresponding genotype model G_{JSP2} and example genotype instance g_6 of the second JSP representation.

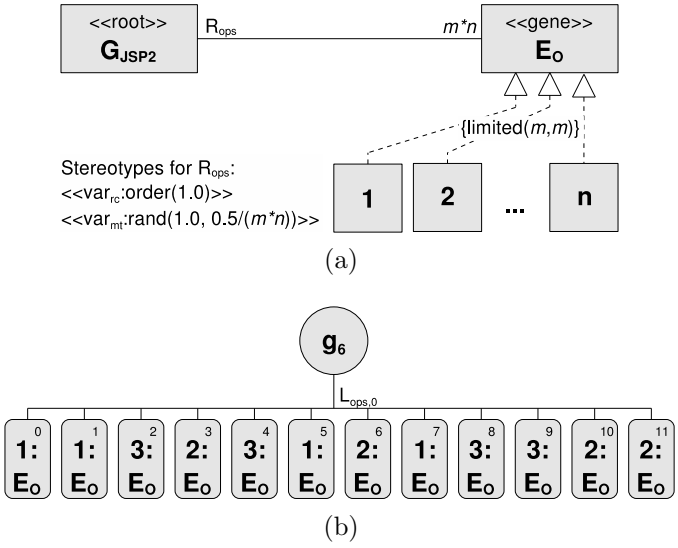


Figure 3.11: Genotype model G_{JSP2} (3.11a) and a possible genotype instance g_6 for $n = 3$ jobs and $m = 4$ machines (3.11b).

To illustrate the decoding mechanism, consider genotype instance g_6 which specifies the following sequence of jobs (denoted by their job ids): 1,1,3,2,3,1,2,1,3,3,2,2. Each job appears exactly four times in the sequence. Each appearance is associated with a corresponding operation for that job: the k-th appearance thus represents the k-th operation for that job. The sequence is decoded into a schedule as follows. The first job in the sequence (i.e., job 1) is considered first. Since this is the first appearance of job 1, it refers more precisely to the first operation of job 1. Therefore, the first operation of job

1 is scheduled on the corresponding machine at time 0. The second job in the sequence is, again, job 1. Since this is the second appearance of job 1, it refers to the second operation of job 1. The second operation of job 1 is scheduled on the corresponding machine as soon as possible. In this case, this means after the first operation is finished. If there is another job processing on the required machine, the operation is scheduled once the machine is available again. Decoding continues in this fashion until all operations of all jobs are scheduled.

3.5.2.3 Evaluation Function f_e and Comparison Function f_c

A genotype g is decoded into a valid schedule and used to determine the makespan $C(g)$. The actual decoding process depends on the representation used. For the two representations the corresponding decoding processes have already been explained in [48] and [10], respectively. The result PI of the evaluation function $f_e(g)$, i.e., the performance of a genotype g, is:

$$PI = f_e(g) = C(g) \tag{3.15}$$

Since the objective is to minimise the makespan, comparison of two JSP genotypes is simple. A genotype g_a is better than another genotype g_b if it results in a schedule that has a shorter makespan:

$$g_a \preceq g_b \iff C(g_a) \leq C(g_b) \tag{3.16}$$

3.5.2.4 Experimental Evaluation

Evaluation is concerned with the comparison of the two representations. For this purpose, the standard problem instance "la32" [81] has been used. A population size of 100 genotypes has been used and the evolutionary process is limited to 500 generations. We evaluate the performance of MEAα in terms of its effectiveness in finding the true

optimum makespan C^{opt} (known from the literature) over a set of 50 experiments as performance metric. For the each experiment, the performance is measured in terms of relative deviation of the best makespan C^{best} found from the true optimum makespan C^{opt}:

$$\delta = \frac{C^{best} - C^{opt}}{C^{opt}} \tag{3.17}$$

For comparison, we consider the mean deviation $\overline{\delta}$ over the entire set of experiments. Figure 3.12 illustrates the convergence behaviour for both representations, using genotype model G_{JSP1} and G_{JSP2}.

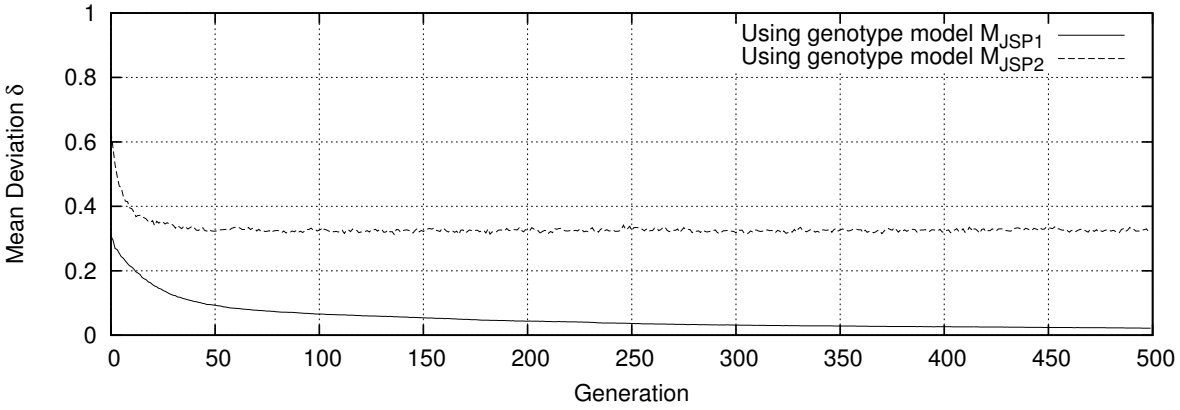


Figure 3.12: Convergence behaviour in terms of mean deviation $\overline{\delta}$ using genotype models G_{JSP1} and G_{JSP2}.

Both representations are indirect representations and need to be decoded into a schedule before the makespan can be determined. The decoding process can influence the performance. ECML is a flexible approach and allows designers to specify different genotype structures that can be used in combination with different decoding methods, i.e., a designer is not limited to a specific pre-defined representation but free to specify a representation that suits the problem best. In our example, we have described two possible genotype models for the JSP. Our results indicate that the more complex structure specified by genotype model G_{JSP1} is more suitable than G_{JSP2}.

3.6 Summary

We have introduced ECML, a formal modelling language for evolutionary computing based on UML. This language can be used to specify genotype structures and their encoding as well as meta information regarding various processing aspects of an EA. In addition, we have introduced the concept of MEAs and described our proof-of-concept implementation of this concept. MEAα is capable of interpreting genotype models specified in ECML and process genotypes instances accordingly. With our approach we effectively separate domain knowledge from the EA implementation. As a consequence, MEAα is more flexible than existing EAs. We have demonstrated the applicability of our approach using various kinds of problems.

In comparison with the existing approaches, ours is closest to grammar-based approaches [65, 106, 130]. However, there are a number of differences. Grammars are used to map genotypes to a syntactically correct sentence which represent the phenotype. Although these phenotypes have a logical tree structure, this information is not preserved in genotypes. For example, grammatical evolution makes use of a binary string representation. In contrast, our approach preserves the logical tree structure in the genotype. In addition, grammar-based approaches have shortcomings with respect to numerical value encoding and crossover points. For this purpose, Nicolau and Dempsey introduced grammar-based extensions [102] that allow for specification of numerical values and crossover points in grammars. Their approach highlights a general problem with grammars which were originally used for language processing where there is no need for numerical values (i.e., numbers are strings of symbols) or recombination. In contrast, ECML explicitly distinguishes between symbolic and numerical alleles. In addition, relationships between the various entities of a genotype model are used as possible recombination points.

ECML can be used to represent problem-specific information Γ in the form of genotype models. In our approach, we distinguish between two kinds of problem-specific information: 1) structural information and 2) processing information. The structural information is used by MEAα to create new genotypes (as explained in Section 3.4.1). It can be used to derive the genotype space $\mathcal{G}$ (i.e., the solution space X). The genotype space comprises of all valid genotypes that can be generated based on a given genotype model. Processing information provided by the genotype model is used by MEAα to process genotypes (as explained in Section 3.4.2 and create new genotypes from existing ones. Furthermore, we have explained that MEAα is used as implementation of f_{opt}. We will use $f_{opt}^{MEA\alpha}$ as part of the symbiotic simulation which will be described in the next chapter. In addition, $f_{opt}^{MEA\alpha}$ will be used for both application showcases that will be subject of Chapter 5 and Chapter 6.

Chapter 4

Agent-based Framework

4.1 Overview

The framework discussed in this chapter is not of theoretical nature but rather focusses on practical issues that have to be addressed when a problem solver agent for a particular application is implemented. Essentially, the framework discussed here is a set of components that are needed to create application-specific problem solver agents. This includes components for integrating the problem solving approach (ECML/MEA) discussed in Chapter 3 and auxiliary components that facilitate the coupling between the physical system and the simulation system. The framework is primarily designed with requirements regarding applicability in mind: the framework must be applicable to different types of symbiotic simulation systems and different application domains, i.e., it must not be limited to a particular application domain such as semiconductor manufacturing, for example. Extensibility is another design feature that has been considered. This might be necessary in the context of some applications where the basic functionality provided by the existing framework components is not sufficient. However, due to the limited scope of this dissertation we focus on the requirement regarding applicability and argue how the proposed design facilitates extensibility.

In this chapter, we discuss existing symbiotic simulation applications regarding their architecture in Section 4.2. This also includes a discussion of two prominent approaches

to implement symbiotic simulation systems: agent-based approaches and Web service-based approaches. We proceed by describing the framework architecture in Section 4.3. This includes an explanation of the various components as well as a discussion of the workflows for class A to C symbiotic simulation systems. We describe the architecture and its various components in an implementation-independent manner first before proposing an agent-based implementation in Section 4.4. In order to show the applicability of the framework we apply it to a number of existing applications that can be found in the literature in Section 4.5. We conclude this chapter with a summary in Section 4.6.

4.2 Background

Various symbiotic simulation applications have been described in the literature. Some of them give details regarding the architecture of the symbiotic simulation system. For example, Low et al. [83] describe a symbiotic simulation system for semiconductor back-end assembly and test operation. Their system is implemented using the JADE agent toolkit [9]. The agent-based system consists of a number of agents which are responsible for system management, monitoring, control, simulation, and optimisation. Kennedy and Theodoropoulos [71] describe an architecture for autonomous DDDAS agents that use symbiotic simulation to predict states of the environment. Based on this prediction, the sensors of the agent can be redirected if necessary. An assistant DDDAS agent is used to compare simulation predictions and real world data to determine whether they are consistent [72]. If not, the simulation model can be revised accordingly. Essentially, a DDDAS agent consists of sensors, a control component, and a model component.

In contrast to the agent-based approaches by Low et al. and Kennedy and Theodoropoulos, a different approach has been chosen by Madey et al. [86, 87] to realise WIPER. The architecture of the system consists of five components: data source, historical data storage, anomaly detection and alert system, simulation and prediction system, and a

decision support system with a web interface [86]. Although the simulation itself is agent-based, the various components of the WIPER system are realised by Web services. Another application of symbiotic simulation is the UAV path planning system described in [67]. Although a detailed description of the architecture is not given, several functional components can be identified from the brief description which can be found in [85]. The symbiotic simulation system consists of several components for sensing and data fusion, scenario creation, what-if simulation, evaluation and analysis, and decision making.

In general, all architectures of the reviewed applications are used for a specific application. To use these application specific architectures for other applications, including other classes of symbiotic simulations, would require to significantly extend their functionality. This is not surprising as none of them was meant to be generally applicable. Each architecture requires a different degree of effort to extend the current functionality. For example, the architecture described by Low et al. [83] can possibly be extended by adding new agents to the system. The WIPER architecture can be extended in a similar way by adding Web services. However, extending the functionality of other architectures seems to be more difficult. For example, the DDDAS agent by Kennedy and Theodoropoulos [71] contains all required functionality for symbiotic simulation anomaly detection. Extending this agent to support other classes would require to significantly enhance the functionality of the DDDAS agent or the development of separate agents for the various classes of symbiotic simulation systems. However, from the information provided it is unclear how modular the original design of the DDDAS agent is.

4.3 Framework Architecture

The various functions (f_{tr}, f_{pa}, f_{opt}, f_{wis}, f_{sim}, f_{pi}, and f_c) that are needed to realise the WIA process have been discussed in Section 2.5. These functions are packaged into a

number of components, provided by the framework. In addition, various other components with corresponding functionality are needed to facilitate the coupling between the physical system and the simulation system. In the remainder of this section, we describe the general architecture of a problem solver agent and discuss the workflows for class A to C symbiotic simulation systems.

4.3.1 Components

We propose a three-layered architecture and corresponding components to realise a problem solver agent. The three layers are responsible for perception, processing, and action. The processing layer consists of the following components: workflow controller component (WORC-C), scenario management component (SCEM-C), simulation management component (SIMM-C), and simulation analysis component (SIMA-C). These components are responsible for performing what-if analyses and thus represent the simulation system. The components of the perception and action layer establish the link between the physical system and the simulation system. The perception layer consists of sensor components (S-Cs). The action layer consists of an action management component (ACTM-C) and actuator components (A-Cs). In this context, we distinguish between internal components, which do not interact with the physical system, and components that facilitate the coupling with the physical system, i.e., sensor and actuator components. Figure 4.1 illustrates an overview of the various components and their respective layers. A description of each component is given below.

4.3.1.1 Sensor Component (S-C)

Sensor components provide sensor data from the physical system to other components of the problem solving agent. We distinguish between three types of data that can be provided by a sensor component: state information (S_t), behaviour information (B_t), and performance indicator information (PPI_t) at a specific time t. A sensor function

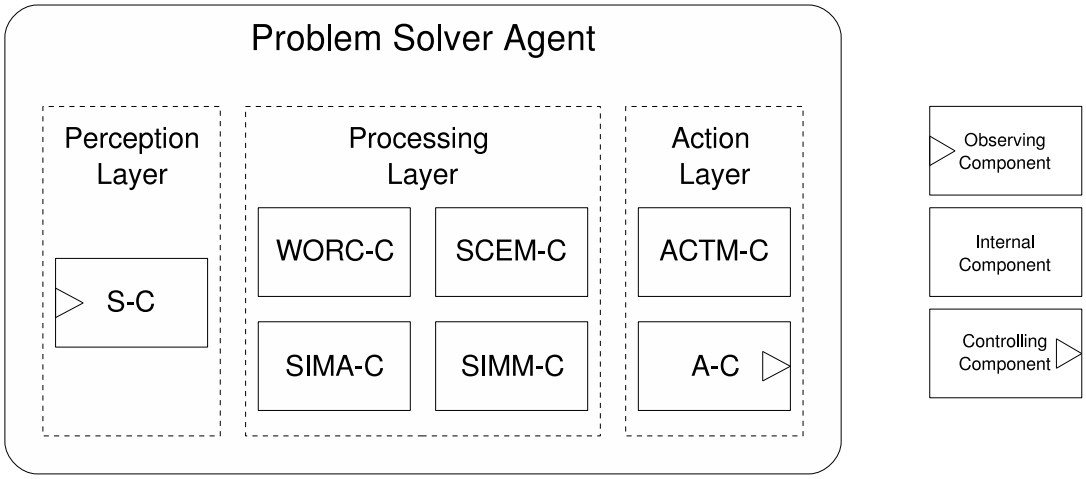


Figure 4.1: Overview of the various components and their respective layers. We distinguish between internal components and components that facilitate the coupling with the physical system, i.e., sensor and actuator components.

$[S_t, B_t, PPI_t] = f_s(t)$ is used for this purpose. Sensors do not have to provide data for all three types of sensor data. For example, there could be multiple sensors, each providing a different type of sensor data. In addition, it is possible that a sensor may provide incomplete data only. For example, there could be multiple sensors that provide state information about the physical system. However, each of these sensors provides data about different aspects of the physical system. The complete state of the physical system can thus only be obtained by aggregating the various state information fragments provided by different sensors. Depending on the type of symbiotic simulation system, PPI_t information may not be available by sensors. For example, in a class B or class C symbiotic simulation system, PI values can only be determined by comparing simulation results with measurements from the physical system. Therefore, PI values cannot be directly provided by sensors. However, in a class A symbiotic simulation system, PI values can be determined based on S_t and B_t by using function f_{pi}. Figure 4.2 illustrates the sensor component.

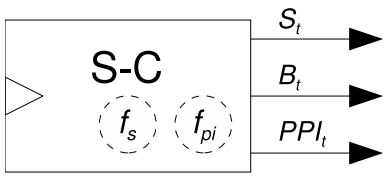


Figure 4.2: Sensor components (S-C) provide data about the state S_t, the behaviour B_t, and performance PPI_t (if applicable) about the physical system.

4.3.1.2 Worflow Controller Component (WORC-C)

The problem solver agent needs to continuously observe the physical system and decide whether it is necessary to perform a what-if analysis according to a class-specific workflow. For this purpose, the WORC-C is periodically evaluating sensor data in order to determine whether or not a WIA process should be triggered. The triggering condition is highly application-specific. For this purpose an application-specific trigger evaluation function f_{tr} is used, which returns either $true$ or $false$, depending on whether a what-if analysis should be triggered or not. Upon triggering a what-if analysis, a pre-WIA is carried out first. As explained in Section 2.5, the pre-WIA process is performed by an application-specific problem analysis function f_{pa} and results in problem-specific information Γ and optimisation objectives Ω. Figure 4.3 illustrates the WORC-C.

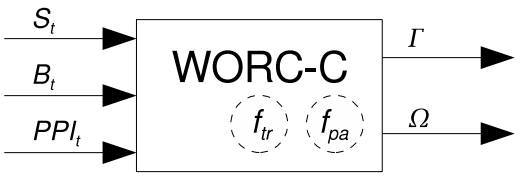


Figure 4.3: The workflow controller component (WORC-C) observes the physical system (S_t, B_t, and PPI_t) and triggers the what-if analysis if necessary (depending on f_{tr}). The pre-WIA process is performed by f_{pa} and results in Γ and Ω values that are needed to perform the actual WIA process.

There are different types of what-if analyses which can be distinguished by their

purpose and by the way how they are triggered. Three types of triggering methods can be found in the literature:

- *Reactive triggering.* The most obvious way of triggering the what-if analysis is the observation of a particular event or condition in the physical system (e.g., [6, 83, 84]. Reactive triggering is the most common form and has been used in various applications. For example, in the context of a back-end assembly and test facility what-if scenarios are simulated in order to find optimal values for upper and lower queue sizes for a particular machine [83]. These queue sizes are required to decide on outsourcing of lots to external vendors. The what-if analysis in this application is triggered if utilisation of machines exceeds a certain limit. Another reactive triggering example is described in [6] which is concerned with control of a set of tools. The what-if scenarios in this application represent alternative tool configurations and the what-if analysis is triggered once the queue length of pending lots has exceeded a certain threshold. Reactive triggering has also been used in the context of an aerospace spare components logistics show-case where a what-if analysis is triggered if the fill rate performance of the physical system falls below an acceptable level [84]. Each what-if scenario uses a different business workflow and the purpose of the what-if analysis is to identify the most suitable one, i.e., the one which produces the best fill rate performance among the evaluated scenarios.

- *Preventive triggering.* Similar to reactive triggering, which depends on observations in the physical system, it is possible to use forecasts to see whether there are likely to be any problems with the physical system in the future. If this is the case, a what-if analysis can be triggered with the goal to either prevent a critical condition or, at least, minimise its negative impact [7]. Since the triggering is based on a forecast, which may be erroneous, there have to be appropriate error handling

strategies to handle two kinds of errors: type 1 errors (a critical condition is not detected) and type 2 errors (a false critical condition is detected). For example, in [7], a reactive triggering method has been used as backup triggering method in case preventive triggering failed.

- *Pro-active triggering.* Another way of triggering the what-if analysis is spontaneous or scheduled instead of being dependent on observations in the physical system. For example, pro-active triggering has been used for UAV path planning [67]. A set of alternative paths is created and evaluated by means of simulation. The purpose of the what-if analysis is to identify the best path for the UAV. Unlike other applications, the notion of a critical condition does not exist and the what-if analysis is triggered periodically. Therefore, the WIA process is triggered in order to continuously improve the performance rather than reacting on a triggering condition in the physical system.

4.3.1.3 Scenario Management Component (SCEM-C)

The WIA process in class A to C symbiotic simulation systems is concerned with optimisation and involves creating a number of candidate solutions to the problem. These candidate solutions represent only a part of the entire what-if scenario. Complete what-if scenarios are composed by combining the candidate solutions, generated by the optimisation function f_{opt} (the implementation used here is $f_{opt}^{MEA\alpha}$, discussed in Section 3.4, which employs MEAα), with additional information. As explained in Section 2.5, composing what-if scenarios is done by f_{wis}. What-if scenario composition also requires a model M and corresponding model parameters PS, PC, and PE. Depending on the class of symbiotic simulation system used, some of these model parameters are provided directly (e.g., provided by sensor data) while others are derived from the solutions provided by the optimisation function:

- *Class A symbiotic simulation system*: the reference model M_R is used, the initial state is provided by the sensors ($PS = S_t$), and PC is derived from candidate solutions created by $f_{opt}^{MEA\alpha}$.

- *Class B symbiotic simulation system*: the model is derived from candidate solutions created by $f_{opt}^{MEA\alpha}$, the initial state is provided by the sensors ($PS = S_t$). Depending on whether the symbiotic simulation system is active or passive, PC is either derived or not used at all.

- *Class C symbiotic simulation system*: the reference model M_R is used, the initial state is derived from candidate solutions created by $f_{opt}^{MEA\alpha}$. Depending on whether the symbiotic simulation system is active or passive, PC is either derived or not used at all.

As explained in Section 2.6, different variations of class B and C symbiotic simulation systems (depending on PC), as well as external influence PE are beyond the scope of this dissertation. For this reason, it is assumed that $PC = \varnothing$ (except for class A symbiotic simulation systems) and $PE = \varnothing$.

The SCEM-C makes use of a comparison function f_c in order to compare the different what-if scenarios with each other. This is important in order to determine the set of optimal what-if scenarios W_o. The WIA process is triggered by the WORC-C which also provides problem-specific information Γ and the optimisation objectives Ω that are required to initialise the optimisation algorithm. For each instance of the WIA process, the optimisation algorithm is initialised with different values for Γ and Ω, depending on the current situation of the physical system and the particular problem at hand. Figure 4.4 illustrates the SCEM-C.

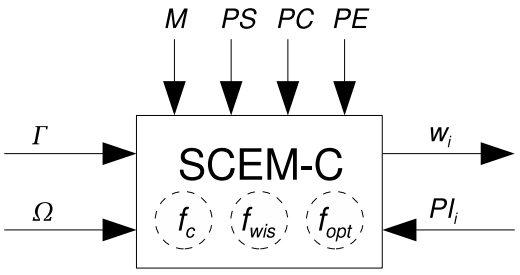


Figure 4.4: The scenario management component (SCEM-C) is using f_{opt} to select solutions from the solution space defined by Γ. Based on a set of solutions X, a corresponding set of what-if scenarios W is composed by f_{wis}. After being simulated, corresponding PI values are available and can be used to compare what-if scenarios with each other. This is done by f_c based on the objectives Ω.

4.3.1.4 Simulation Management Component (SIMM-C)

The various what-if scenarios created by the SCEM-C are simulated by the SIMM-C using the simulation function f_{sim}, discussed in Section 2.5. Since simulation models are highly application-specific, the SIMM-C has to be specially provided for each application. The SIMM-C thus represents the link between the problem solver agent and the application-specific simulation engine. Simulation results of a what-if scenario w_i are provided as state and behaviour information $[\mathbf{SS}, \mathbf{SB}]_i$ and/or performance indicator information PI_i (if available). Depending on the application, not all types of information may be provided. For example, in a class A symbiotic simulation system, the only data that is required for performing the WIA process is PI information that enables comparison of what-if scenarios in terms of their fitness to solve the problem. Figure 4.5 illustrates the SIMM-C.

The SIMM-C implementation may contain a simulation engine or, alternatively, may employ a suitable 3rd party simulation engine. In general, there are no limitations regarding the use of 3rd party simulation engines as long as there is a way of feeding scenarios to the simulation engine and retrieving corresponding simulation results, respectively. For

Figure 4.5: The simulation management component (SIMM-C) is using f_{sim} to simulate a what-if scenario w_i. Depending on the class of symbiotic simulation system (i.e., class A or class B/C) and the application-specific simulation engine, simulation results are either provided as state and behaviour information $[\mathbf{SS}, \mathbf{SB}]_i$ or performance indicator information PI_i.

example, the simulation engine may be a COTS simulation package such as AutoSched[7] or Arena[8]. It is also possible to use a simulation engine that has been specially developed in Java, C/C++, or any other programming language. However, since every simulation engine has a different interface, regardless whether it is a 3rd party software package or not, it is necessary to provide for a mechanism that allows the SIMM-C to communicate with the simulation engine. This requires the SIMM-C implementation to make sure that information provided by what-if scenarios is translated into a form that can be interpreted by the simulation engine. Similarly, it is necessary to convert simulation results into a form that is compatible with the framework.

From the problem solver agent's perspective, performing simulations is transparent, i.e., the SIMM-C is a black-box that accepts input in form of what-if scenarios and provides corresponding output in form of $[\mathbf{SS}, \mathbf{SB}]$ and/or PI. Therefore, the SIMM-C is also responsible for handling issues concerned with stochastic simulation models. This kind of model require the execution of multiple simulation replications of the same what-if scenario in order to collect enough data to derive a statistically sound summary of the results. This typically involves, but is not limited to, the determination of means and standard deviations of the corresponding data. An important issue in this context

[7] `http://www.appliedmaterials.com/services-software/library/autosched-ap`
[8] `http://www.arenasimulation.com/`

is also the required number of simulation replications. Depending on the application and the simulation model, an appropriate number of simulation replications has to be used. However, in the context of symbiotic simulation it may not be possible to define a reasonable number of replications at design time. A simulation of the physical system may exhibit time and situation-dependent stochastic behaviour, i.e., depending on the time or the current situation of the physical system, simulation results may exhibit more or less variance. There is various work in the literature that provides methods to optimise the number of required simulation replication in order to save computing budget. For example, Chen and Lee discuss an optimal computing budget allocation method [18].

4.3.1.5 Simulation Analysis Component (SIMA-C)

The WIA process in class A to C symbiotic simulation systems is concerned with optimisation. As explained in Section 2.5, the optimisation process requires one or more performance indicators that are used to compare what-if scenarios with each other. In case the simulation of what-if scenarios does not directly result in PI information, the simulation results have to be further analysed. This is done by the SIMA-C using an analysis function f_{pi}. For example, in the case of a class B or class C symbiotic simulation system, PI information can only be obtained by comparing simulation results $[\mathbf{SS}, \mathbf{SB}]_i$ with the measurements S_t and B_t from the physical system. In contrast, in a class A symbiotic simulation system for instance, the PI information can usually be determined directly by the simulation and further analysis of the simulation results is not necessary. However, depending on the application, it might be necessary to further analyse the simulation results even for class A symbiotic simulation systems. Therefore, the SIMA-C is optional. Figure 4.6 illustrates the SIMA-C.

Figure 4.6: The simulation analysis component (SIMA-C) analyses simulation results $[\mathbf{SS}, \mathbf{SB}]_i$ using f_{pi} and determines corresponding performance indicator values PI_i. Depending on the class of symbiotic simulation system, this may also involve using state and behaviour information (S_t and B_t, respectively) from the physical system.

4.3.1.6 Action Management Component (ACTM-C)

Depending on the symbiotic simulation system, the problem solver agent is exercising control over the physical system. The optimisation problem solved by the WIA process may have several solutions. In case of a class A symbiotic simulation system, this essentially means that there is a number of possible alternative decisions that represent solutions to the problem. A single decision has to be propagated to the physical system. This is done by the ACTM-C using an action selection function $PC = f_{as}(W_o)$. Upon completion of a WIA process, the set of optimal what-if scenarios W_o is forwarded to the ACTM-C which is responsible for making the final selection based on a corresponding policy. The control parameters PC of the selected what-if scenario is then forwarded to the various actuator components in the system. Alternatively, the ACTM-C may also reject the entire set of solutions. For example, this may be the case if none of the solutions is satisfying certain requirements). Figure 4.7 illustrates the ACTM-C.

Figure 4.7: The action management component (ACTM-C) selects a what-if scenario from W_o using f_{as} and extracts the corresponding PC parameter values.

4.3.1.7 Actuator Component (A-C)

An A-C is responsible for implementing given control parameters PC into the physical system by using an actuator function $f_a(PC)$. There can be multiple actuator components in a symbiotic simulation, depending on the application-specific requirements. Figure 4.8 illustrates the A-C.

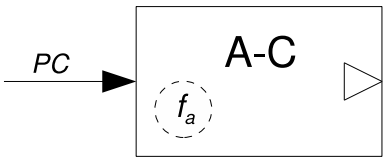


Figure 4.8: Actuator components (A-C) implement PC into the physical system using f_a.

4.3.2 Workflows

The workflows for class A to C symbiotic simulation systems are both concerned with solving optimisation problems and are thus very similar. The main difference is due to the need to compare simulation results with sensor data from the physical system to determine performance indicator values in class B and class C symbiotic simulation systems. In addition, class B and class C symbiotic simulation systems do not control the physical system and, thus, do not require components of the action layer. In this section, we discuss the workflows based on the various framework components.

Class A Workflow The purpose of a class A symbiotic simulation system is to evaluate a number of decision alternatives and implement the best solution into the physical system. The WORC-C is continuously observing the physical system. If a triggering condition is satisfied (i.e., if $f_{tr} = true$), then the what-if analysis is started. First, the current situation (i.e., the current operational conditions of the physical system) is analysed by the pre-WIA process (f_{pa}), resulting in a problem statement in terms of Γ and

Ω. Next, the actual WIA process is started in order to find solutions to the problem. The WIA process is primarily performed by the SCEM-C and the SIMM-C in combination with the SIMA-C (if required). The SCEM-C composes what-if scenarios that are evaluated by the SIMM-C. Comparison of what-if scenarios is based on performance indicator values which are either directly provided by the simulation results or have to be determined by the SIMA-C. Once the optimisation process is finished, the found solutions W_o are analysed by the ACTM-C to determine the actual action (in terms of control parameters PC) that is being implemented in the physical system. The selected action is forwarded to the actuator components. The workflow of a class A symbiotic simulation system is illustrated in Figure 4.9.

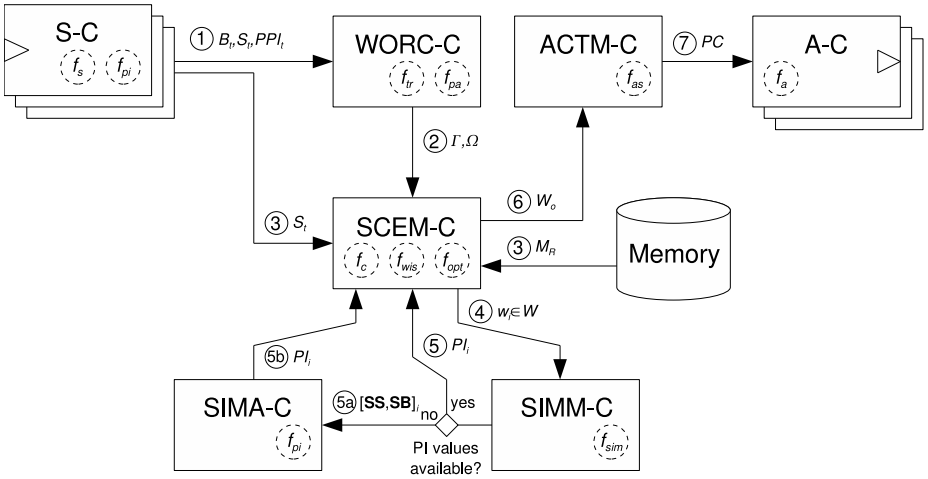


Figure 4.9: Workflow for class A symbiotic simulation systems.

Class B and Class C Workflow Similar to the workflow of a class A symbiotic simulation system, the workflows for class B and class C symbiotic simulation systems, involves the WORC-C to observe the physical system, analyse the situation, state the problem, and trigger the what-if analysis to find a solution. Optimisation in class B and class C symbiotic simulation systems is similar as both are concerned with minimising the

deviation between observations from the physical system and the simulation. However, there are some differences. The SCEM-C creates what-if scenarios that are concerned with different models (class B symbiotic simulation system) or assumptions regarding the state (class C symbiotic simulation system) of the physical system. The simulation results of each what-if scenario is compared with the sensor data from the physical system by the SIMA-C in order to determine the performance indicator values. This is different from class A symbiotic simulation systems where performance indicator values can be determined without the need to compare it with sensor data. Problem solving in class B and class C symbiotic simulation systems is not concerned with controlling the physical system. Solutions are thus not implemented in the physical system using actuators. Instead, solutions are kept in memory. The workflow of a class B and a class C symbiotic simulation systems is illustrated in Figure 4.10.

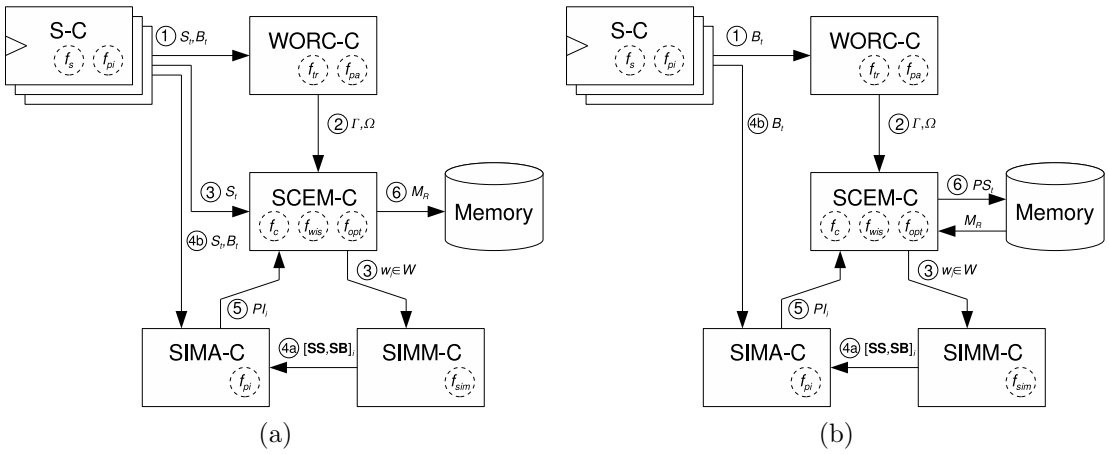


Figure 4.10: Workflow for a class B symbiotic simulation systems (4.10a) and a class C symbiotic simulation system (4.10b).

4.4 Framework Implementation

We propose an agent-oriented infrastructure for realising symbiotic simulation-based problem solver agents. As far as terminology is concerned we need to clearly distin-

guish between the problem solver agent on the higher, conceptual level and the software agents on the lower, technical level. The problem solver agent, which has so far been the subject of the dissertation, is a system that can be considered an agent by itself as it is capable of autonomously reasoning and interacting with its environment (i.e., with the physical system). However, so far we have only discussed the problem solver agent on a conceptual level, i.e., we have not discussed any implementation-specific details. This section will be concerned with the implementation-specific details and explain how to use low-level software agents to realise a higher level problem solver agent.

4.4.1 Capability-centric Design

Our approach is based on agent systems and the concept of capabilities [13, 15]. The concept of capabilities is used in agent systems to facilitate modularisation of functionality. Busetta et al. introduced the concept of capabilities and described a capability as "... a cluster of plans, beliefs, events, and scoping rules over them" [15]. A similar concept of capabilities was described in Sabater et al. [120] using the term 'module'. The original concept was later extended by Braubach et. al, who explained that "... the capability concept addresses most of the five fundamental criteria of modularization from [96]: decomposability, composability, understandability, continuity and protection". The extended capability concept is realised in Jadex [113].

Modularisation is an important issue in satisfying the requirements regarding applicability and extensibility of the proposed framework. Depending on the application context and the type of symbiotic simulation system used, some functionality may not be required. For example, a class A symbiotic simulation system does not necessarily require a SIMA-C in order to determine the PI values for the simulation results. Another example is that the components of the action layer are only required by those symbiotic simulation systems that influence the physical system. For constructing an application-specific problem solver agent, it is therefore necessary to be able to tailor the design

of the problem solver agent to the requirements of the application and the symbiotic simulation system. A modular approach to providing functionality is thus important. The capabilities concept provides the required degree of modularisation. Therefore, our approach to designing a problem solver agent is capability-centric.

Depending on the application, the problem solver agent has to provide a certain functionality which is provided by the various components that have been introduced in Section 4.3.1. Every framework component is realised as a capability. Software agents equipped with a particular capability possess the functionality provided by the capability. For example, any software agent that is equipped with a simulation capability is capable of simulating what-if scenarios. Software agents can be equipped with an arbitrary number of capabilities. It is therefore possible to have a single software agent which is equipped with all capabilities required to realise the entire problem solver agent. On the other hand, the same problem solver agent can be realised by using multiple software agents. In this case every software agent is equipped with some of the capabilities required by the system. We thus distinguish between a multi-agent system and a single-agent system as illustrated in Figure 4.11.

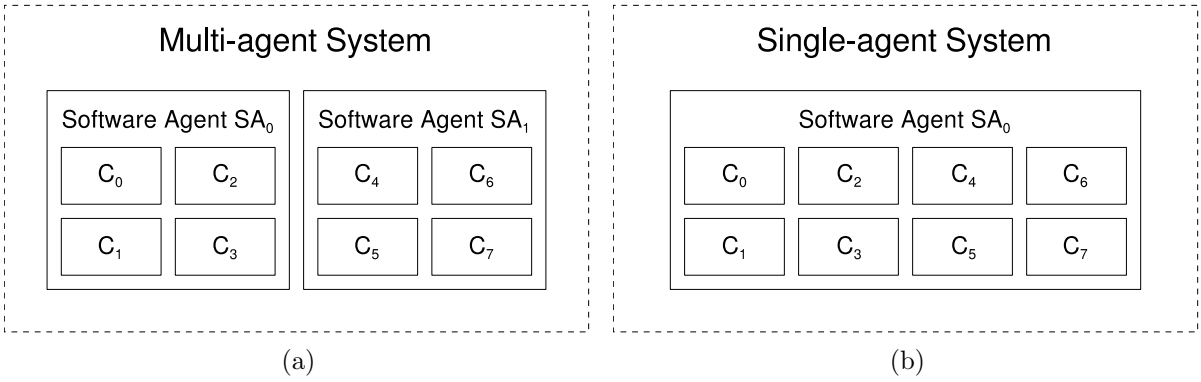


Figure 4.11: Comparison between multi-agent system (4.11a) and single-agent system (4.11b).

A problem solver agent can be realised in either way, i.e., a problem solver agent can be realised by one or more software agents, each equipped with capabilities needed in order

to realise its complete functionality. Designing the problem solver agent as multi-agent system may be necessary for various reasons. For example, a physical system may be inherently distributed and sensors have to be placed at various locations. In this case, it might be necessary to deploy various software agents, each equipped with a corresponding sensor capability. In addition, simulations may be very compute intensive and are thus performed by an off-site high-performance computing cluster. This may require a software agent, equipped with a simulation capability, to be located on the cluster. Whether or not a distributed problem solver agent design is required depends on the application. The capability-centric approach described here does not dictate a particular style. Instead, it enables the designer to chose a design that suits the application.

4.4.2 Capabilities

Each of the various components of the framework is realised as a capability and consists of two parts: a core and an interface. The core implements the actual functionality of the component and the interface allows other capabilities to access the functionality of the core. For example, the function that allows the SIMM-C to simulate what-if scenarios (i.e., f_{sim}) is implemented by a corresponding core. Capabilities communicate with each other either by using function calls or by passing messages. While message passing is always an option, function calls are only possible if the capabilities are provided by the same software agent. Communication using function calls is preferable because it does not involve any communication overhead as it is the case with message passing. Figure 4.12 illustrates the general structure of a capability.

Most cores are inherently application-specific. For example, the core of the SIMM-C is responsible for simulating the various what-if scenarios. Different applications use different simulation engines and may require different computing environments for performing the simulation. Therefore, the SIMM-C core implementation is inherently application-specific and different core implementations have to be provided for each application.

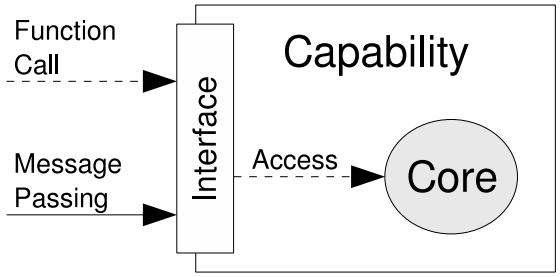


Figure 4.12: Overview of the general structure of a framework capability.

However, depending on the similarity of different applications, it might be possible to re-use some of the application-specific core implementations.

In contrast to application-specific cores, the optimisation core used by the SCEM-C can be realised in a more generic way. In Chapter 3 we have introduced our approach of using a high-level modelling language (ECML) in combination with an appropriate meta algorithm (MEAα) to separate problem-specific information from the algorithm implementation. We have applied this approach to different problems, showing the re-usability of our approach. The optimisation core of the SCEM-C can thus be re-used for different applications. An overview of the various framework components is illustrated in Table 4.1. This table also indicates whether a component is inherently application-specific or more generic. Although it is possible to use an application-specific optimisation core (if necessary), we use the same core implementation (i.e., MEAα) throughout this dissertation.

4.4.3 Service Registry and Service Discovery

In the context of the framework, components are generally considered as service providers, i.e., a component provides certain types of services. For example, the SIMM-C capability provides a service which simulates what-if scenarios and returns simulation results (i.e., data regarding state and behaviour, i.e., $[\mathbf{SS}, \mathbf{SB}]$, or performance indicators PI). As

Table 4.1: Overview of the various framework capabilities, indicating whether their core implementations is inherently application-specific (A), or more generally applicable (G).

Capability	Function	Purpose	A/G
S-C	f_s	Provide access to sensor data	A
WORC-C	f_{tr}	Trigger what-if analyses	A
WORC-C	f_{pa}	Perform pre-WIA process	A
SCEM-C	f_{opt}	Perform optimisation	G
SCEM-C	f_{wis}	Compose what-if scenarios	A
SCEM-C	f_c	Compare what-if scenarios	A
SIMM-C	f_{sim}	Simulate what-if scenarios	A
SIMA-C	f_{pi}	Determine PI values	A
ACTM-C	f_{as}	Select action to be implemented	A
A-C	f_a	Control physical system	A

mentioned above, components are realised as agent capabilities and since software agents can be equipped with multiple capabilities, they can provide an arbitrary number of services. Each software agent is equipped with a registry capability that keeps track of the services provided by its various capabilities. In addition, the same registry also maintains a list of services provided by other software agents. For this purpose, registry capabilities inform other registries about their services. Figure 4.13 illustrates the registration and discovery of services.

4.5 Application Examples

In this section, we explain how the framework could be used to realise existing symbiotic simulation applications. For this purpose, we describe three conceptual showcases, each based on an application found in the literature. The purpose of these conceptual showcases is to show that the framework architecture fits a variety of applications. Actual proof-of-concept applications will be the subject of Chapters 5 and 6. The three applications, concerned with UAV path planning, semiconductor manufacturing, and a wireless emergency response system, have been chosen based on the availability of details

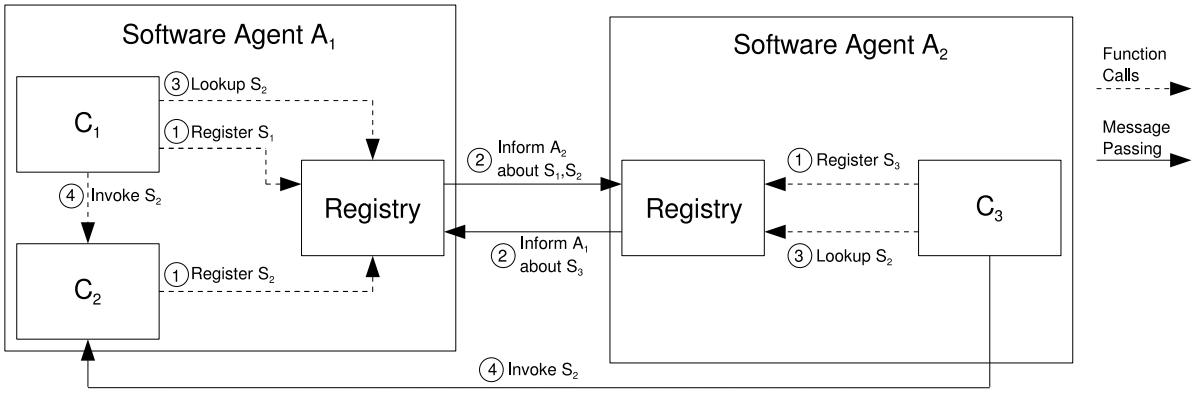


Figure 4.13: Overview of service registration and discovery: capabilities register their services with the local registry capability (1), registry capabilities of different agents synchronise each other (2), capabilities lookup services (3) and invoke services offered by another capability (4).

regarding their architecture and design which makes it easier to see how the symbiotic simulation framework could be applied.

4.5.1 UAV Path Planning

The symbiotic simulation system described in [67] is used for adaptive path planning of UAVs. Information about the location of the UAV and the target is provided by corresponding sensors. Alternative what-if scenarios are created based on a set of alternative paths and simulated concurrently. Each scenario is initialised with the current location of the UAV and *a priori* estimations about the target. The simulation results are compared in order to determine the best path, i.e., the path that minimises the estimated detection time of the target. Detailed algorithms for path planning and simulations are given in [67]. The authors mention that it is possible to apply the results directly to the UAV, i.e., implying a control system with actuators. An illustration of the original architecture for this application can be found in the technical report on symbiotic simulation-based decision support in [85]. Figure 4.14 illustrates how the UAV path planning application can be realised using the various components of the proposed framework.

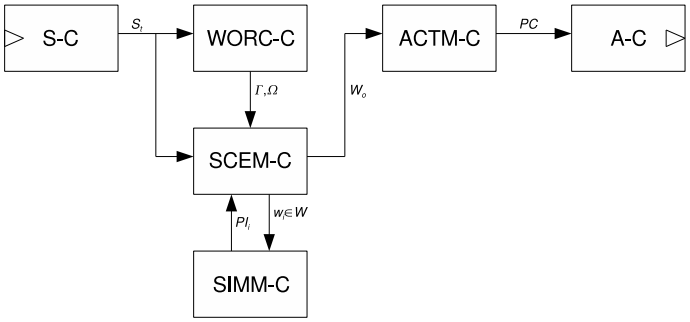


Figure 4.14: Design of a problem solver agent for UAV path planning, based on the proposed framework.

Sensors (S-C) provide the data concerned with the state of the physical system. For example, this includes information regarding geography of the road network and current location of the UAV. The WORC-C periodically triggers the what-if analysis. A set of what-if scenarios W, each concerned with an alternative UAV path, is created by the SCEM-C, based on *a priori* problem-specific information Γ about the target. The objective Ω of the optimisation process is always to minimise the estimated detection time, regardless the current situation. Each scenario w_i is simulated by the SIMM-C. A SIMA-C is not required as the simulation results PI_i directly reflect the estimated detection time. The best what-if scenario W_o, identified by the WIA process, is propagated to the ACTM-C and implemented into the UAV using actuators (A-C).

4.5.2 Semiconductor Manufacturing

The symbiotic simulation system described in [83] is used to optimise out-sourcing decisions by considering the operating cost for a semiconductor back-end factory. To avoid late deliveries and consequent penalties, wafer lots are marked for out-sourcing under certain conditions. Marking is switched on/off if the number of lots in a queue exceeds or falls below specified upper or lower limit, respectively. An optimal combination of upper and lower queue sizes yield minimum operating cost. In this application, the purpose

of the WIA process is to find an optimal value for these queue sizes. Sensors are used to retrieve the current state of the factory. This information is used to initialise the simulations. In addition, each what-if scenario uses a different set of upper and lower limits (i.e., different decision making alternatives). The result of the what-if analysis can be directly used in the physical system to make decisions regarding out-sourcing. Figure 4.15 illustrates how the application can be realised using the various components of the proposed framework.

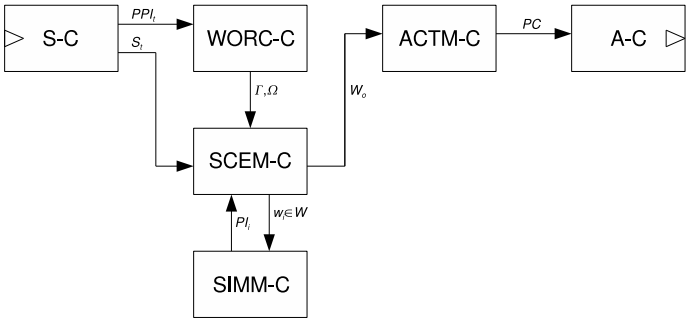


Figure 4.15: Design of a problem solver agent for back-end operations in semiconductor manufacturing, based on the proposed framework.

A sensor capability (S-C) provides performance indicator information (PPI_t) about the machine utilisation in the physical system at time t. If the utilisation exceeds 80%, a what-if analysis is triggered. The objective Ω of the what-if analysis is always to minimise the estimated operating cost. The symbiotic simulation system described by Low et al. does not make use of any form of situation-dependent problem-specific information, i.e., $\Gamma = \varnothing$. As a result the optimisation problem remains exactly the same for all subsequent what-if analyses. During the WIA process, a number of what-if scenarios W are created by the SCEM-C and simulated by the SIMM-C. Each what-if scenario w_i is concerned with different values regarding the upper and lower queue limits. The current state of the physical system (S) is used as initial state for the simulations. The simulation result PI_i indicates the expected operating cost and no additional analysis is required, i.e., no

SIMA-C is required. The best what-if scenario W_o is propagated to the ACTM-C which uses the actuator A-C to implement the new settings PC in the physical system.

4.5.3 Wireless Emergency Response System

WIPER monitors communication patterns of millions of cell phone users in order to recognise anomalies, possibly indicating emergency situations. Sensor data includes activity data (e.g., call initiator and recipient) and spatial location data provided by mobile phone companies, as well as the 30 second call detail record tags which identify the closest cell tower to the phone [87]. WIPER consists of multiple sub-systems. An anomaly detection subsystem (ADS) is responsible for detecting anomalies by comparing sensor data with historical data of normal communication and spatial location patterns. Once an anomaly is detected, the simulation system is triggered. This simulation system is responsible for classification of the detected anomaly by continuously validating simulations by comparing their predictions with sensor data. Once validated, a simulation is used to predict how the anomaly is likely to unfold in the near future. This information is also shared with the decision support system (DSS) which is responsible for activating various emergency response applications that can be used, for example, to disseminate various information to emergency crews, engineers, and managers.

Although WIPER consists of multiple sub-systems, some of which could possible be realised as a symbiotic simulation system, we are only concerned with the anomaly classification subsystem for this conceptual showcase as it is an example for a class B symbiotic simulation system. The ADS could possibly be realised with a class D symbiotic simulation system. However, this is a Group 2 symbiotic simulation system and thus not within the scope of this dissertation. Anomalies are classified by testing various hypotheses about the nature of the anomaly. The one that describes the anomaly best is used to predict its further evolution. The class B symbiotic simulation system described

in [86], as part of WIPER, is used to classify anomalies by testing various hypotheses. The design for a corresponding problem solver agent is is illustrated in Figure 4.16. The ADS and DSS are illustrated as external components, respectively.

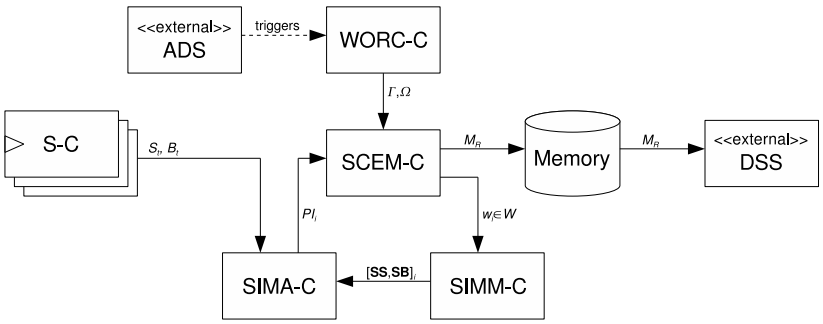


Figure 4.16: Design of a problem solver agent for anomaly classification in the context of WIPER, based on the proposed framework.

In this example, we discuss the case where the anomaly classification system is realised by a class B symbiotic simulation system. Sensor capabilities (S-C) provide information such as the location of the cell phone (included in S) and cell phone activity (included in B). Once an anomaly is detected by the ADS, the WORC-C triggers a what-if analysis. A set of what-if scenarios W is created by the SCEM-C. Each what-if scenario represents an alternative hypothesis (realised as different models) regarding the anomaly (e.g., traffic jam) and is simulated by the SIMM-C. The SIMA-C validates a what-if scenario w_i by comparing simulation output $[\mathbf{SS}, \mathbf{SB}]_i$ with sensor data (S_t and B_t) from the physical system. The result of this validation task is a performance indicator value PI_i which represents a quantitative assessment regarding how closely the simulation results match the observations in the physical system. The objective Ω is to minimise the discrepancy between the simulation and the physical system. Problem-specific information Γ may include *a priori* information regarding the cause of the anomaly. The model M of the best what-if scenario, i.e., the what-if scenario that represents the most plausible hypothesis for the behaviour of the physical system, is used as reference model $\mathcal{M}^R = \{M\}$. The

reference model is kept by the problem solver agent and shared with the other subsystems of WIPER for further use. For example, the DSS can make use of the reference model to evaluate how the event is likely to unfold in the near future.

4.6 Summary

Although the various classes of symbiotic simulation systems have a distinct purpose, they are composed of only a few common functionalities. We have explained the functions that are needed to realise a WIA process in Section 2.5. In this chapter we have introduced additional functions (i.e., f_s and f_a) which are responsible for linking the simulation system with the physical system. Based on the various functions we have proposed an agent-based architecture which uses the concept of capabilities to realise corresponding components. Requirements regarding applicability and extensibility of the framework are achieved by modularisation using capabilities. This allows users of the framework to add new capabilities (if necessary) and design application-specific symbiotic simulation systems by using only those capabilities that are needed. In addition, each capability can be reasonably customised by providing application-specific core implementations that are needed to simulate or compare what-if scenarios, for example.

We have discussed a number of conceptual applications that illustrate how existing applications could be realised using the framework. Not all of these applications are explicitly identified as symbiotic simulation systems by their creators. For example, the WIPER application is introduced as a DDDAS. However, parts of this application make use of the same concepts that are central to symbiotic simulation. There are more examples in the literature that use symbiotic simulation concepts. However, applying the framework to all of them is beyond the scope of this dissertation. A smaller proof-of-concept showcase in the context of semiconductor manufacturing [6] has been realised with an early version of the agent-based implementation of the framework. In addition,

another symbiotic simulation application, concerned with a supply chain [140], has also been realised using the framework. In the context of this dissertation, we discuss two major showcases in Chapter 5 and Chapter 6.

Chapter 5

Application Example 1: Semiconductor Manufacturing

5.1 Overview

Semiconductor manufacturing is a complex process which turns pure silicon wafers into integrated circuits with thousands of components. The whole process may require up to several hundred processing steps and up to three months for production [114]. One crucial factor for competitiveness in semiconductor manufacturing is ongoing improvement of the manufacturing process [114]. Automation is a critical factor in this context and the semiconductor industry invests several million US dollars for solutions. According to [47], investments for integrated automation solutions can range between US$ 130 million and US$ 180 million. These integrated solutions also include systems for real-time control of equipment. In addition to high complexity, the semiconductor manufacturing process is also an asset intensive process. A variety of tools are used in semiconductor manufacturing to process wafers and a single tool can cost up to US$ 2 million [121]. Improvement of tool efficiency is therefore important in order to reduce cost for additional tools.

Several performance metrics are considered in semiconductor manufacturing, of which throughput, cycle time, and on-time delivery are considered the most important ones [111].

The performance of most tools depends on their operation mode, further referred to as configuration, and the distribution of different products, further referred to as product mix. If the product mix is changing, it might be necessary to reconfigure various tools in order to maintain performance targets. Decisions regarding the tool configuration are made by engineers who estimate the required capacity for each product mix. These decisions are then implemented by operators who manually change the configuration of a tool. However, operators do not always strictly follow their instructions. This, and the fact that decision making relies mostly on the experience of engineers, causes the demand for an automated control solution.

Scheduling of wafer lots to production resources (e.g., tools) is a crucial performance issue in semiconductor manufacturing. A variety of work has been concerned with this issue. For example, a large body of work is concerned with the evaluation of dispatching policies. These policies decide, according to some criteria, in which sequence wafer lots are being assigned to production resources. Research in this area often focuses on a single type of tool, such as stepper tools [135] or furnaces [2], for instance. In this context, the work by Zhang et al. [141] is notable as it employs the same concepts that are essential to symbiotic simulation. They describe a system in which a scheduler is monitoring and controlling a semiconductor manufacturing plant (fab). The scheduler determines when a new rule combination (dispatching rule + rework strategy) has to be used. Rule optimisation is done with the support of a simulator that is employed by the scheduler to evaluate the performance of a given combination of dispatching rule and rework strategy. Once the best combination has been identified, it is implemented in the fab. Essentially, what Zhang et al. describe is a class A symbiotic simulation system.

Dispatching (i.e., assignment of a wafer lot to a tool) is dependent on the current setup of a tool. A wafer lot can only be dispatched if the setup of a particular tool is compatible with the one needed by the wafer lot. Changing tool setups can be time

consuming and frequent re-configurations should thus be avoided in order to reduce inefficient tool utilisation. This can be achieved by employing dispatching policies that explicitly consider setups [123]. However, while policies can be used to locally optimise the performance of parts of the fab, they are not well suited to optimise the entire fab. A fab is a highly complex system and changes to one part of the system are likely to affect other parts. In order to evaluate how changes to one part of the system will affect other parts, simulations are required. Policies that do not make use of simulations are thus not well suited for optimisation of an entire fab. In contrast, the concepts of symbiotic simulation can be used to improve the semiconductor manufacturing process of an entire fab.

In this chapter, we apply our symbiotic simulation-based approach to a semiconductor manufacturing system in order to improve throughput by scheduling the change of setups more efficiently. A problem solver agent is used in combination with a common practice approach for local decision making. While rule-based local decision making is sufficient for some parts of the manufacturing system, the problem solver agent identifies and focuses on problematic parts. Existing work, such as the one by Zhang et al., often aims at changing setups of tools only as a result of dispatching wafer lots to production resources that requires a different setup. Before the lot can be processed, the setup needs to be changed. In contrast, our approach directly decides on the schedule for changing setups. Any decision regarding the change of setups by the problem solver agent overrides the local decision making process. As a result, our approach is highly adaptive as the problem solver agent is dynamically changing its focus of attention as the operating conditions are changing. This is another major difference to related work, which lacks this sort of adaptability.

The advantage of our approach is the ability to predict the effects of changing setups on the overall performance of the fab. In addition, our approach is not limited to a number

of predefined policies. Instead, the problem solver agent has the freedom to change the setup of any tool at any point of time. The disadvantage of this very high degree of freedom is an extremely large number of possible decisions. We show how situation-specific what-if analysis can be performed by using the approach described in the previous chapters. In the context of this application, we show that our approach improves the performance of the physical system in terms of throughput and the performance of the WIA process in terms of computing efficiency.

This chapter is structured as follows. In Section 5.2 we discuss the model of the semiconductor manufacturing system. In Section 5.3 we discuss the design of the problem solver agent used for this application. In addition, we discuss how the WIA process can be specialised, depending on the current operating conditions. In Section 5.4 we discuss a number of experiments with focus on the impact of using a problem solver agent to complement local decision making and the impact of specialisation on the performance of the WIA process. We conclude this chapter by discussing the application example and summarise the contribution of this chapter in the context of the dissertation in Section 5.5 and Section 5.6, respectively.

5.2 Semiconductor Manufacturing

The model of the semiconductor manufacturing system considered in this application is based on a real-world fab located in Singapore. It has been developed with the support from domain experts at D-SIMLAB Technologies[9]. Although information about the real-world fab has been used to develop the model, it does not exactly reflect the real fab for proprietary reasons. The semiconductor manufacturing business is highly competitive and detailed information regarding the various processes and the manufacturing environment represents a valuable asset to manufacturing companies. Knowledge about

[9] `http://www.d-simlab.com`

these details are thus kept strictly confidential. Some details have not been available for developing the model. For example, this includes the exact number of tools as well as precise timings of the various processing steps.

The purpose of this application is to demonstrate the applicability of our approach to realise a problem solver agent, based on a class A symbiotic simulation system, for decision support in semiconductor manufacturing. In particular, the advantage of situation-dependent problem solving in comparison with common practice operational decision making is being investigated. For this purpose, it is important to have a model which realistically reflects the complexity of the semiconductor manufacturing process. It does not necessarily have to exactly reflect any existing fab with all its proprietary details. Therefore, for creating the model, we have focused on a realistic degree of complexity rather than the exact representation of a particular real-world fab.

5.2.1 Manufacturing Process

A fab can produce multiple products concurrently. From the perspective of the manufacturing process, products are characterised by the different processing steps that are needed to turn the raw wafer into the finished product. A total of 18 different products are considered in this application. Each product is associated with a workflow that represents a sequence of processing steps. The workflow indicates the exact order in which wafers have to be processed, the tool and setup that have to be used for each processing step, as well as the exact processing times needed to complete a step. The total number of processing steps required by various products ranges between 169 and 347. On average, a product has to go through 252 different steps during its manufacturing process. The time required for a wafer to complete the entire manufacturing process is commonly referred to as cycle time. We distinguish between the theoretical cycle time, which is the total processing time of all steps without any delays, and the actual cycle

Table 5.1: Overview of the various products, including the number of processing steps and the theoretical cycle time for each product.

Product Id	# of Steps	Theoretical Cycle Time	Product Id	# of Steps	Theoretical Cycle Time
P0001	169	7.6 days	P0010	239	8.9 days
P0002	240	8.8 days	P0011	239	8.9 days
P0003	347	14.8 days	P0012	334	14.8 days
P0004	347	14.8 days	P0013	233	9.0 days
P0005	345	14.8 days	P0014	196	7.4 days
P0006	334	14.8 days	P0015	196	7.4 days
P0007	236	10.1 days	P0016	233	9.0 days
P0008	345	14.8 days	P0017	206	7.8 days
P0009	240	8.8 days	P0018	225	3.4 days

time, which also includes delays caused by transportation within the fab, for example, or unavailability of processing resources. The average theoretical cycle time is 9.8 days. The shortest and longest theoretical cycle time is 3.4 days and 14.8 days, respectively. An overview of the various products is illustrated in Table 5.1.

A wafer is a circular slice of a semiconductor material such as silicon (Si) or gallium arsenide (GaAs). The diameter and thickness of wafers varies. Typical sizes used in industry range from a diameter of 50.5mm with a thickness of 275μm to a diameter of 300mm with a thickness of 775μm. Wafers are kept in cassettes for easier handling. Typical cassettes used for 300mm/775μm wafers can hold up to 25 wafers. Depending on the processing step, wafers are processed either one by one or by the lot, i.e, an entire cassette at once. Some tools also allow batch processing, i.e., to process multiple wafer lots concurrently. Although wafers might be processed one by one, they are always transported from one tool to another in their cassettes. Loading and unloading of wafers from their cassette is done fully automatically by the corresponding robotic handlers in the tool. This reduces the need of human intervention which would otherwise lead to impaired quality due to improper handling. In the simulation model, we do not explicitly represent wafers. Instead, the smallest entity are wafer lots and for the remainder of this

Table 5.2: Overview of the various tool types, number of station families, total number of stations in the fab model, and the maximum batch size in terms of wafer lots (if the tool allows batching).

Tool Type	# of Families	# of Stations	Batching
Backside Clean	1	2	N/A
Cluster tool	23	108	N/A
CMP tool	3	5	N/A
DNS Clean	3	3	N/A
DUV	1	5	N/A
Furnace	10	33	6 lots
Hardbake	1	2	6 lots
Hotplate	1	2	6 lots
I-Line	1	15	N/A
Implanter	3	21	N/A
Laser Marker	1	2	N/A
Metrology	6	52	N/A
Wetbench	11	36	2 lots

chapter, we will use wafer lots (with exactly 25 wafers) as the unit for denoting workload and throughput.

Various kinds of tools are used in the semiconductor manufacturing process. This includes tools such as furnaces, lithography tools, cluster tools, wetbenches, and a number of other tools. The fab model considered in this application has a total number of 286 stations, each of which is equipped with a single tool. They are organised into 65 station families. Each station family has a storage capacity for temporarily keeping wafer lots until a station becomes available for processing waiting lots. Depending on the tool, processing is done either by lots or batches of lots. In case, a tool is capable of processing wafer lots in batches, corresponding batching policies are used that determine how batches are composed. An overview of the various tool types used in this application is illustrated in Table 5.2.

Tools can be operated with different setups. However, although a tool may technically be capable of operating under a certain setup, it may not be certified for this purpose.

Certification is an important measure to maintain the required quality of wafers. The certification procedure can last several weeks and is thus not subject to operational decision making which is usually in terms of hours and days. Therefore, from this application's perspective, certifications cannot be changed. The number of setups supported by a certain tool varies greatly. While some tools are certified to support up to 27 different setups, other tools support only a single setup. The average number of certified setups per tool is 6. Changing the setup of a tool often requires downtime (i.e., setup time) during which the tool is not available for processing any wafers.

Not all tools require setup times. For example, in case of wetbenches, a "setup" refers to a particular recipe according to which wafers are treated. Wetbenches are tools that are used to clean the wafer after certain fabrication steps to remove residues which would otherwise affect the quality of the wafer. This cleaning process requires wafers to be processed in a number of chemical liquids for a precisely specified time period. A recipe is thus a unique sequence of chemicals and specific processing times. Different wafers can be processed according to different recipes (i.e., different chemical sequences and timings). However, a wetbench is equipped with a fixed set of chemical baths. Although the liquids have to be replaced from time to time (maintenance), they are not changed (i.e., replaced by a different chemical). Therefore, there is no downtime (i.e., setup time) required when processing wafers according to a different recipe.

In contrast, furnace tools may operate at different temperatures (in this case "setup" refers to a temperature setting). Wafers have to be processed at a certain temperature for a certain period of time. To assure quality, it is important that the temperature is not fluctuating. Therefore, if wafers require different processing temperatures (i.e., different setups), a furnace needs to change its operating temperature. The ramp up/down (heating up/cooling down) process takes some time (i.e., the setup time) during which no wafers can be processed. An overview of the various setup times of all tool types (if applicable) is illustrated in Table 5.3.

Table 5.3: Overview of tool types and their corresponding setup times in minutes. Actual setup times are assumed to follow a normal distribution. Therefore, corresponding mean setup times and standard deviations are given.

Tool Type	Setup Time Mean (Std.Dev.)	Tool Type	Setup Time Mean (Std.Dev.)
Backside clean	15 (5)	Furnace	35 (8)
Cluster tool	15 (5)	Hardbake	15 (5)
DNS clean	15 (5)	Hotplate	15 (5)
DUV	20 (3)	I-Line	20 (3)

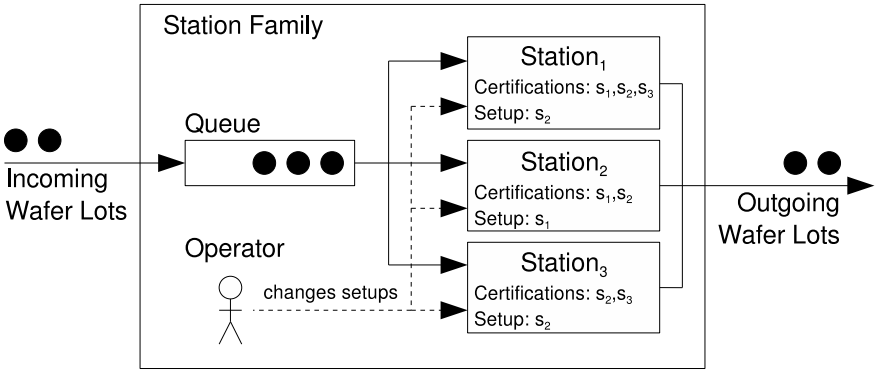


Figure 5.1: Example of a station family with three stations, each of which is certified for different setups.

Despite efforts towards automated operations, operational decisions regarding configuration of tools is made by human operators. Each station family is assigned an operator who is ultimately responsible for changing the setups of the various supervised stations. Figure 5.1 illustrates an example of a station family with three stations, each of which is equipped for different setups.

An overview of the complete process graph is illustrated in Figure 5.2. Each node in the process graph represents a different station family. Arrows indicate the path of a wafer lot through the entire manufacturing process. Lots may revisit any station family multiple times. The figure shows the graph for all products. It is not meant for detailed reference but rather to illustrate the complexity of the manufacturing process.

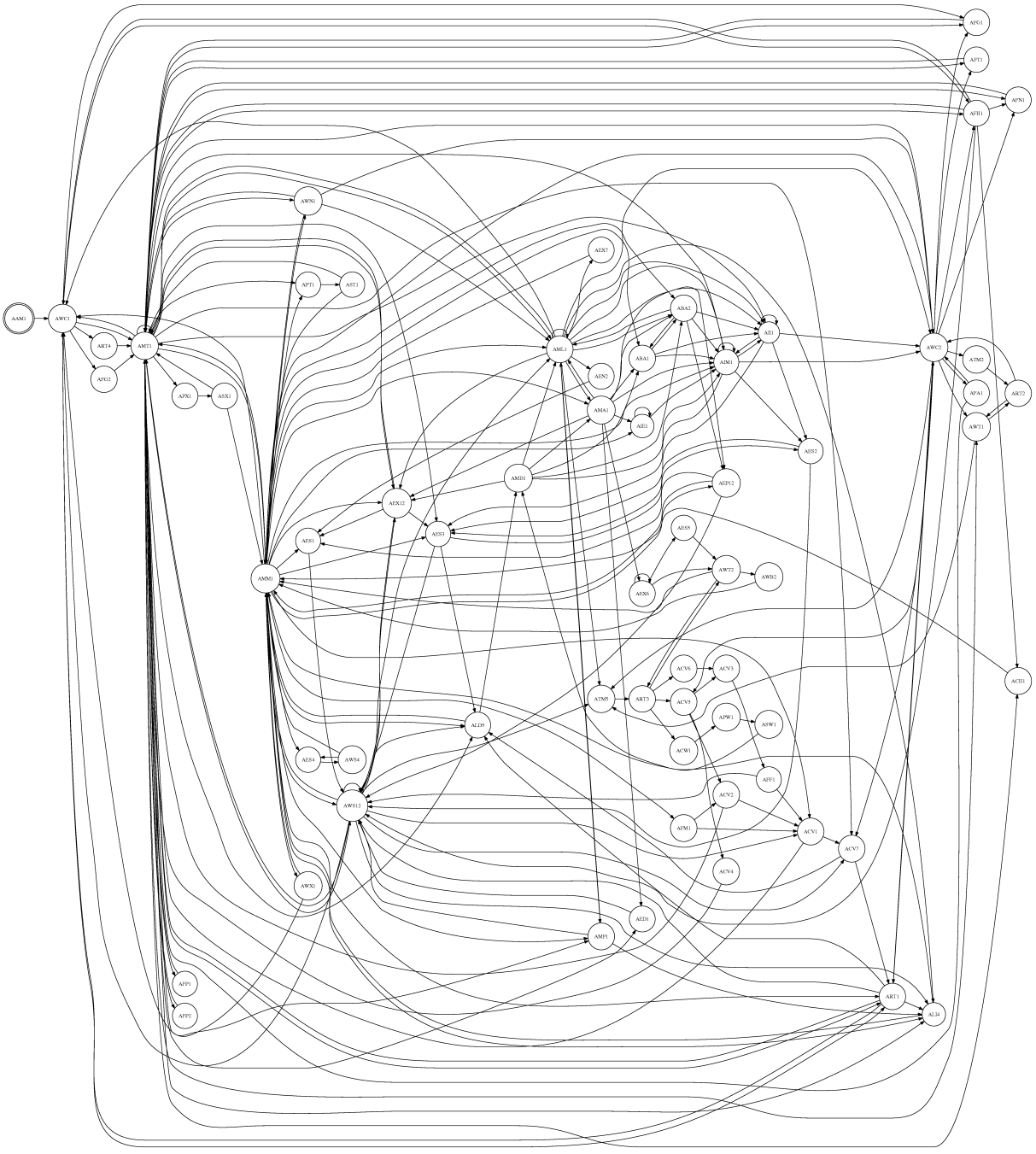


Figure 5.2: Complete process graph of the semiconductor manufacturing process, including all products and station families. Each node in this graph represents a station family. Note, this figure is not meant for detailed reference but rather to illustrate the complexity of the manufacturing process.

5.2.2 Decision Making

Common practice decision making is distributed (i.e., operators make decisions regarding the stations under their supervision) and involves local optimisation (i.e., each operator tries to optimise the performance locally in terms of throughput, for example). Although operators of different station families can communicate with each other and coordinate their actions, they cannot fully anticipate how local decision making is going to affect different parts of the fab. This is due to the complexity of the manufacturing process. Simulations are required to see how local decision making affects other parts of the fab. Therefore, centralised decision making, using full knowledge about the state of the fab, is beyond the capabilities of human operators.

The local decision making policy used in our model is based on information provided by the industry regarding the operation of stations and their re-configuration. It is generally distinguished between priority and non-priority stations. Priority stations are the ones that are more critical to the manufacturing process and refer to stations that are equipped with cluster tools, DUV tools, furnaces, implanters, and wetbenches. Operators watch priority stations more closely and act faster upon changing operational conditions as compared to non-priority stations. The setup of non-priority station is assumed to be changed only once for every working shift (i.e., once every 8 hours). In contrast, priority stations may be re-configured every hour if the situation requires. Operators can analyse the setup demand of the waiting lots in the queues of the station families supervised by them. For our model we assume that the goal of the operator is to match setup demand and supply by re-configuring as many stations as necessary in order to minimise the discrepancy between the setup demand and supply.

A simple way of calculating the demand of a setup would be to count the number of lots in the queue that require this particular setup for their next processing step. However, this measure does not take into account the processing time required by this

step. The processing time depends on the processing step and varies among different products and their specific process flows. The lot count does not accurately reflect the demand situation as many lots with short processing time would outweigh few lots with long processing times. Demand in terms of number of lots is thus misleading. Hence, we consider demand in terms of the waiting processing time. The waiting processing time $wpt(s)$ for a particular setup s is the accumulated processing time of all lots in the queue that require the setup. The total waiting processing time WPT for a station family is thus the accumulated waiting processing time for all setups that are provided by the various stations of the station family. In order to match demand and supply, we further consider the demand ratio $DR(s)$ and the supply ratio $SR(s)$ of a particular setup s which represent the setup demand/supply relative to the total demand/supply, respectively. The demand ratio $DR(s)$ for setup s can be calculated as:

$$DR(s) = \frac{wpt(s)}{WPT} \tag{5.1}$$

While setup demand is measured in waiting processing time, setup supply is measured in number of tools $nt(s)$ that are configured for the corresponding setup s. Based on the total number of tools NT within a station family, the supply ratio $SR(s)$ for a specific setup s can be calculated as:

$$SR(s) = \frac{nt(s)}{NT} \tag{5.2}$$

In order to match demand and supply, the objective of the operator is to minimise the discrepancy δ_{DS} between demand and supply, which is defined as:

$$\delta_{DS} = \sum_{s \in C} |DR(s) - SR(s)| \tag{5.3}$$

where C is the set of all setups that are provided by the various stations of the station family.

Operators are assumed to be free in their choice of selecting tools within a station family for changing setups. Therefore, every time an operator checks the current operating situation of a station family, multiple tools may be selected for re-configuration in order to match demand and supply. In a fab it is often common practice to use a "fire fighting" approach. In our model, we model this common practice approach as follows:

1. Determine the critical setup for each station family, i.e., the setup which is highest in demand compared to its supply.

2. Identify all stations within the station family that are certified for this setup but currently use a different setup.

3. Select one of the identified stations and change the setup. This results in a lower demand/supply discrepancy.

This process is repeated until no further improvements in terms of demand/supply discrepancy are possible.

5.3 Problem Solver Agent

The disadvantage of a common practice decision making policy, such as the one described in the previous section, is the lack of complete knowledge and the ability to anticipate how decisions regarding one part of the system affect other parts. Without the use of simulation, operations are limited to local optimisation based on local knowledge about the situation within a single station family. In case of human operators, these limitations are not necessarily that severe. For example, human operators have knowledge about the system and are thus capable of anticipating, to a certain extent, the likely outcome of their actions. This is particularly true for operators who can make use of experience with similar situations and problems in the past. In addition, operators can communicate

with each other and coordinate their actions. Nevertheless, due to the complexity of the system, it can be assumed that operators have always only partial knowledge about the system. In contrast, a symbiotic simulation-based problem solver agent has the advantage of being able to exploit a detailed model of the physical system to make fully informed decisions using complete knowledge about the state of the physical system.

The purpose of the problem solver agent in this application is to override local decision making, whenever necessary, to improve the efficiency of the manufacturing process. For this purpose, we assume that the various stations of the physical system are primarily controlled by the operators who make decisions according to the common practice approach discussed in Section 5.2.2. In addition, a problem solver agent is used to identify problematic station families and optimises the schedule for changing setups of the various stations. Since the problem solver agent is concerned with decision making problems, a class A symbiotic simulation system is used. In this application, we consider the case where the problem solver agent is concerned with continuously improving the overall performance of the manufacturing process. As we will discuss below, the complexity of the decision making problem for the entire fab is very high. We therefore aim at improving the performance of the WIA process by dynamically changing the attention of the problem solver agent to focus on the most problematic station families. Figure 5.3 illustrates the design of the problem solver agent for the semiconductor manufacturing application.

The S-C and A-C represent the link between the problem solver agent and the physical system. In practice, a fab is equipped with an information management systems that keeps track of the whereabouts and progress of the various wafer lots in the fab. The problem solver agent described in this chapter is capable of autonomous and automated operational decision making regarding changing the setups of the various stations. This agent relies heavily on the availability of accurate information regarding the state of

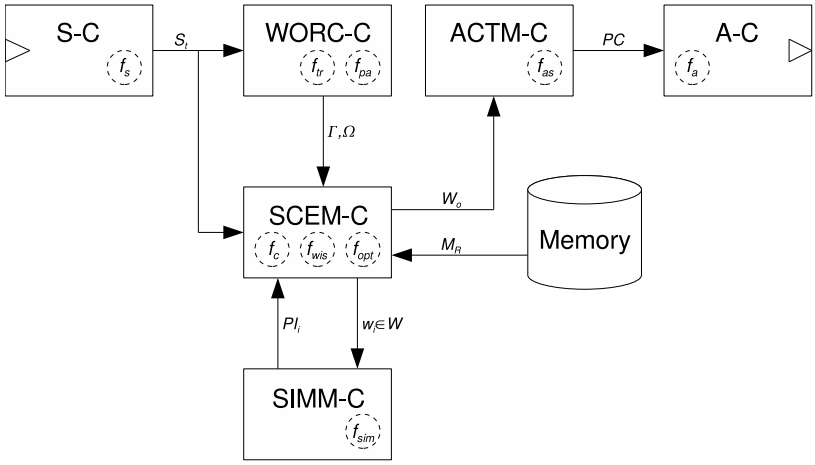


Figure 5.3: Design of the problem solver agent, based on a class A symbiotic simulation system, for control of semiconductor manufacturing equipment.

the physical system. In addition, although decision making can be done by the problem solver agent, re-configuration of stations in a real-world fab probably has to be performed by human operators who manually change the tool setups. The actuator of the problem solver agent is thus assumed to be a human-computer interface that allows the agent to send corresponding instructions to the human operators in the fab.

In this application, WIA processes are performed pro-actively: a what-if analysis is triggered every 12 hours and concerned with a time period that covers the next 24 hours. It is assumed that at any time t, the information management system of the physical system provides information about the complete state S_t of the physical system. The pre-WIA process analyses the current state of the physical system and determines problem-specific information Γ and objectives Ω depending on the current operating conditions. Here, operating conditions refers to the queueing situation of the various station families in terms of waiting processing time. The remainder of this section is concerned with details regarding Γ and Ω.

5.3.1 Problem-specific Information Γ

The genotype model used to perform the optimisation process is based on the various stations and their possible setups that are being considered by the WIA process. In general, the genotype model reflects stations as genes, denoted as E_i, and their setups as corresponding alleles, denoted as $s_{i,j}$, where i and j refer to the index of a particular station and setup, respectively. A gene E_i which is expressed by an allele $s_{i,j}$ thus indicates that the current setup of the i-th station is changed to the j-th setup. It may not be necessary or desired to change the setup of a particular station at all. For this purpose, we consider an additional allele nil_i for each station which indicates that the current setup is not changed. Therefore, for gene E_i, representing the i-th station, there are exactly $ns_i + 1$ alleles, where ns_i refers to the number of possible setups for the i-th station.

The goal of the WIA process is to identify an optimised schedule for changing setups of the various tools in the physical system. We assume that the setup of a tool can be changed at most once an hour. Given a time frame of 24 hours (i.e., a what-if analysis covers a 24 hour period), there are a total of 23 time slots at which the setup of a station can be changed. This takes into account the fact that the what-if analysis needs some time to finish. In this application, we assume that the WIA process can be performed in less than an hour. Time slots are represented in the genotype model by using the multiplicity attribute of the associative relationships R_i that connect the root element with genes. Therefore, there is a total of 23 instances of a gene in a genotype instance and the k-th instance of the i-th gene refers to the setup that needs to be applied to the i-th station during the k-th time slot.

Genotype instances generated from this genotype model encode instructions for operators when to change the setup of a particular station and which setup to use. Although the structural information in the genotype model takes into consideration that setups

do not necessarily have to be changed (by providing *nil* alleles), it does not account for the frequency of changing setups throughout a 24 hour period. Some stations are more frequently changed than others. In practice, setups are not changed every hour as this would cause a significant overhead due to the setup times incurred. Based on observations with the common practice decision making approach, setups are (on average) changed 5.8 times during a 24 hour period. The probability of *not* changing the setup of a station during any given time slot, on average, is thus 75.8%. We use a customised allele probability distribution to reflect these observations. The allele probability for selecting the *nil* allele is thus 0.75 and the corresponding allele probability $p_{i,j}$, for selecting a setup allele $s_{i,j}$, is $0.25/ns_i$.

The genotype model, featuring the various structural and advanced processing attributes we just described, is further denoted as G_{FAB} and illustrated in Figure 5.4. This genotype model also specifies the various variation operators that have to be used by MEAα to perform recombination and mutation operations. For this application, we use a standard uniform crossover operation (indicated by `<<var_rc:uniform(1.0, 0.5)>>`) with a crossover probability of $v_0 = 0.5$. In addition, we specify that a random mutation operator (indicated by `<<var_mt:rand(1.0, 1/m)>>`). In both cases, the operators are applied with a probability of $p_{var} = 1.0$. The mutation operator has an additional parameter $v_0 = 1/m$ that indicates the mutation rate used by the operator. The mutation rate is inversely proportional to the number of stations that being considered (i.e., m). The operators, as well as the corresponding probabilities, have been chosen based on experience.

Based on the number of stations (m) and the corresponding number of setups for each station $(ns_0, ns_1, \ldots, ns_{m-1})$, the size of the solution space defined by genotype model G_{FAB} can be calculated as follows:

$$\prod_{i=0}^{m-1} (ns_i + 1)^{23} \tag{5.4}$$

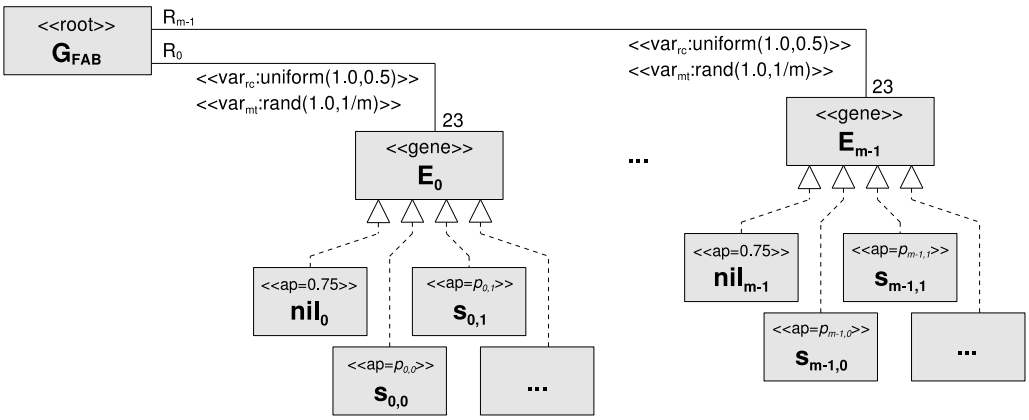


Figure 5.4: General genotype model G_{FAB} for the setup scheduling problem in semiconductor manufacturing.

In this most general case, the genotype model considers all stations and all their possible setups (plus one additional *nil* allele). Given the large number of stations and possible setups, the size of the solution space is approximately 4×10^{3313}, which can be safely described as extremely large. A smaller size of the solution space makes it easier for the search algorithm to find a good solution. For this reason, we use two methods to significantly reduce the size of the solution space based on a case-by-case analysis of operational conditions of the physical system.

1. Attention of the problem solver agent focuses on problematic station families only and ignores less problematic ones. A watch list, consisting of problematic station families, is created based on a forecast of the waiting processing times for all station in the next 24 hours. A station family is considered problematic if its waiting processing time is above the P-th percentile of the waiting processing times of all station families. The degree of specialisation (i.e., focus) is further denoted by $P = \{0, 50, 75, 95\}$, where $P = 0$ indicates that the problem solver agent has no focus, i.e., problem solver agent considers all station families.

2. Setups that are not needed (i.e., setups for which the forecast has not indicated any demand) are ignored and not represented in the genotype model by the corresponding alleles. For this purpose, the queue of the station family is analysed. In case all setups of a particular station can be ignored, the entire station is ignored and not included in the genotype model. Whether or not this setup filter is applied or not is further denoted by $filter = \{yes, no\}$.

5.3.2 Objectives Ω

For the objectives we assume an application scenario in which the manufacturer is ultimately interested in improving the fab throughput, i.e., to increase the number of wafer lots that can be completed during a period of time. A simple way of measuring the throughput would be to count the number of wafer lots that finish processing during that time period. However, this way of measuring throughput has a serious drawback that may lead to performance problems. The manufacturing process requires wafer lots to visit many stations multiple times. Simply considering the number of lots that finish processing will lead to short sighted behaviour of the problem solver agent because it will configure stations in way that will give priority to wafer lots that are nearly finished. Although this will (briefly) increase the number of lots that finish in their processing, the performance improvement is superficial. It is achieved at the expense of wafer lots that are in an earlier processing stage which will have to wait.

In order to avoid this issue, we consider the average throughput of the fab. Let wip_i denote the i-th progress group, i.e., the set of wafer lots that have completed processing at least $10 * i$ percent and less than $10 * (i + 1)$ percent. For example, wip_4 denotes the progress group that consists of all wafer lots with a progress of 40%–50%. Furthermore, let tp_i denote the throughput of wafer lots from the i-th progress group to their successive progress group $(i + 1)$. For example, tp_4 would denote the throughput of wafer lots from

progress group 4 to progress group 5. The average throughput $\overline{tp}$ of the fab can now be determined as follows:

$$\overline{tp} = \frac{\sum_{i=0}^{9} tp_i}{10} \tag{5.5}$$

The primary objective of the WIA process is to maximise the average throughput $\overline{tp}$. However, improved throughput is a side effect of reduced inefficiencies in the manufacturing process. Inefficiencies of a particular station family is expressed by the corresponding waiting processing time for this station family. Therefore, the secondary objective of the WIA process is to minimise the waiting processing time WPT for the various station families that are subject to optimisation (i.e., station families that are on the watch list). Since the pre-WIA process determines the watch list every time a what-if analysis is triggered, Ω is different for every what-if analyses. For example, while the what-if analysis at one point of time may be concerned with minimising the waiting processing time of furnaces; the what-if analysis at a different time may be concerned with minimising the waiting processing time of cluster tools and DUV tools.

Both objectives are used by the search method employed by the problem solver agent. For this purpose, MEAα has been modified in order to distinguish between primary and secondary objectives as follows:

- The secondary objective is used for parent selection and offspring creation. In other words, the secondary objective is used to evolve the population of candidate solutions during the problem solving process.

- The primary objective is used to determine the final solution by comparing all candidate solutions that have been created during the WIA process according to the primary objective.

In the context of this application this means that candidate solutions are evolved according to WPT, i.e., solutions with shorter waiting processing times for the various

station families are considered better, and the final solution is determined according to $\overline{tp}$, i.e., the candidate solution with the highest average throughput is selected to be implemented in the physical system. Although the average throughput is not directly used to drive evolution of candidate solutions, it is effectively improved because of the above-mentioned relationship between average throughput and inefficiencies.

5.4 Experiments

5.4.1 Impact of Problem Solver Agent

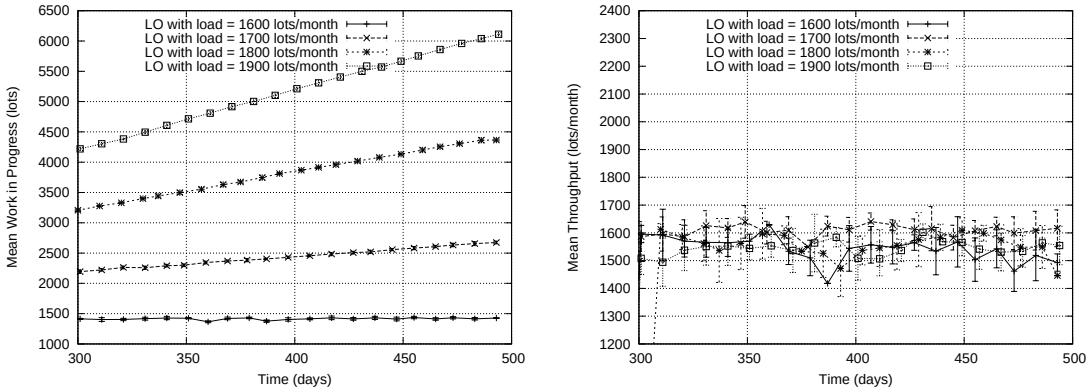
We compare the performance of the physical system when using common practice approach for decision making (i.e., no symbiotic simulation) and the symbiotic simulation-based approach using problem solver agent. The purpose of this comparison is to determine the maximum load the physical system can handle depending on the decision making approach used. As for problem solving by the problem solver agent, we do not consider the effects of problem-specific knowledge at this point (i.e., $P = 0$ and $filter = false$). Therefore, for each what-if analysis that is conducted by the problem solver agent, the same (generic) problem is solved. This generic problem is concerned with all stations and all possible setups, i.e., with the entire search space. As explained earlier in Section 5.3, a what-if analysis is triggered every 12 hours and simulations concern the next 24 hours. Each WIA process is given a computing budget of 48000 simulations. In addition, each what-if scenario is evaluated using 3 simulation replications. This results in a maximum of 16000 what-if scenarios that can be evaluated during every WIA process.

For practical reasons, experiments were not performed with a real physical system. Instead, we used a paced simulation of the fab to emulate the physical system. This enabled us to perform experiments, reflecting a period of several months, within a few weeks. We also used a mechanism to create snapshots of the complete state of the emulator at any time. These snapshots have been used as initial state for the simulation

of a what-if scenario. The simulation model is deterministic with the only exception being the setup times. In practice, the exact setup times are not known *a priori*. Therefore, all simulations performed by the WIA process use different random number generator seeds to produce different values for the various setup times according to a normal distributions (see Table 5.3 for details regarding the setup times).

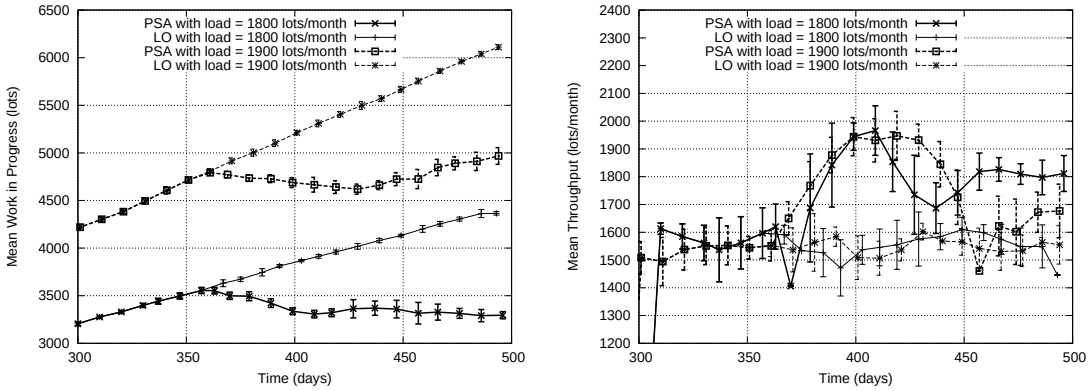
For evaluation purposes, we consider the WIP and the average throughput of the physical system. The WIP is used to determine whether the decision making approach is capable of handling a certain amount work load. More specifically, we consider work loads between 1600 and 1900 lots per day for local decision making and 1800 and 1900 lots per day for the problem solver agent. A set of five experiments is performed for each load and decision making approach. The results presented below represent the corresponding averages (for WIP and throughput) over these five replications. As for the conduct of experiments, each experiment has been started from scratch, i.e., the semiconductor manufacturing system is empty with no WIP. Wafer lots are released daily according to the corresponding work load. Figures 5.5 and 5.6 show the performance of the physical system when using the common practice approach and when using the problem solver agent. The problem solver agent is only activated at day 360. In addition, both figures only show the performance starting from day 300. The period before this day is considered the warm-up period and thus not illustrated.

When using the common practice approach, the maximum load that can be handled by the physical system is approximately 1600 lots per month. At higher loads, the physical system cannot keep up which results in a steadily increasing WIP (see Figure 5.5a). The upper work load limit is also indicated by the throughput which is approximately between 1500 and 1600 lots per month regardless the load. In order to handle a given work load, the physical system needs to be able to maintain a throughput which equals the load. Clearly, this is not the case for work loads above 1600 lots per month when



(a) (b)

Figure 5.5: Performance of the semiconductor manufacturing system when using local decision making only (LO): WIP (5.5a) and throughput (5.5b).



(a) (b)

Figure 5.6: Performance of the semiconductor manufacturing system when using the PSA: WIP (5.6a) and throughput (5.6b). For easier comparison, corresponding results for local decision making only (LO) is also illustrated.

using the common practice approach. In contrast, when using the problem solver agent, higher work loads can be handled. This is indicated by a WIP that does not significantly increase or decrease (see Figure 5.6a). When using the problem solver agent, the physical system can maintain a throughput of up to 1800 lots per month (see Figure 5.6b).

Initially, after activating the problem solver agent, throughput increases sharply and even exceeds the load. For example, in Figure 5.6b, throughput reaches almost 2000 lots per month around day 400 which is significantly higher than the load of 1800 or 1900 lots per month used for these experiments. This is due to the fact that, before the problem solver agent was activated on day 360, only common practice decision making has been used. Since the common practice approach cannot handle work loads of 1800 and 1900 lots per month, excess WIP has been accumulating in the queues of the physical system. This excess WIP is eventually processed when the problem solver agent is activated. As a result, when using a load of 1800 lots per month, the throughput converges to approximately the same level. However, even the problem solver agent cannot handle a load of 1900 lots per month and throughput remains significantly below this level which explains the increasing WIP (see Figure 5.6a).

5.4.2 Impact of Problem-specific Knowledge

We evaluate the impact of problem-specific knowledge on the performance of the WIA process using different degrees of specialisation, expressed by values for P in combination with or without using a setup filter. Unlike the previous experiments, which were concerned with the performance of the physical system, we consider the performance of the WIA process in terms of computing budget needed to achieve a certain quality of solutions. As explained earlier in Section 5.3.2, the primary objective of a what-if analysis is to maximise the average throughput. The secondary interest is to minimise the waiting processing time of stations under control by the problem solver agent. For

this reason, quality of solutions is measured in terms of these two performance metrics. An important feature of our approach is the ability to perform situation-specific what-if analyses, i.e., the current operating conditions of the physical system are taken into consideration during the pre-WIA process when defining the problem (in terms of Γ and Ω). We hypothesise that a specialised WIA process performs better (i.e., achieves higher quality of solutions with the same amount of computing budget) as compared to a generic WIA process.

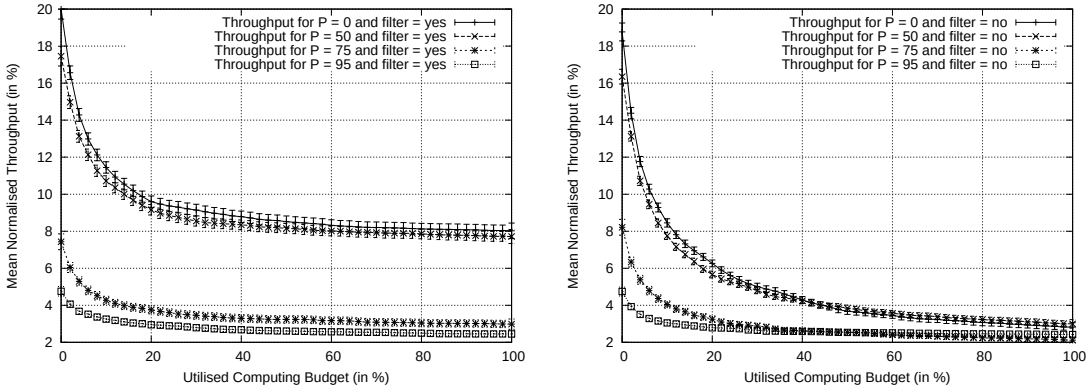
When using a specialised WIA process, the specific problem that is being solved depends on the current operational conditions of the physical system. What-if analyses performed at different times are thus likely to be concerned with different problems (e.g., the what-if analyses may be concerned with different station families). Thus, corresponding solutions of these what-if analyses cannot be directly compared. However, in order to show the impact of specialisation it is not sufficient to analyse the performance of the WIA process for a single what-if analysis only. Instead, the average performance over a number of what-if analyses should be considered. For this reason, a set of experiments is performed. Each experiment is performed using a different operational condition based on a snapshot of the physical system. More specifically, a snapshot is the saved state S_t of the physical system at time t. Snapshots have been obtained from the emulator of one of the experiments, using a load of 1800 lots per month, discussed in the previous section. A total of 35 snapshots is considered, each of which has been taken at different days (ranging from day 360 to 395).

For each snapshot, we analyse the performance of the WIA process for all combinations of $P = \{0, 50, 75, 95\}$ with $filter = \{yes, no\}$. For each experiment the same parameters regarding computing budget (48000) and simulation replications (3) have been used as for the experiments discussed in the previous section. Similarly, each WIA process has been repeated 3 times to obtain statistically meaningful results. Therefore,

for each of the 35 snapshots a total of $|P| \times |filter| \times 3 = 4 \times 2 \times 3 = 24$ experiments is conducted. Since results obtained for different snapshots cannot be directly compared, we define a normalised performance measure which considers the performance of the WIA process relative to a reference performance. For this purpose, we determine the best performance (maximum throughput and minimum waiting processing time) that has been obtained for each snapshot over the set of 24 experiments and use it as reference performances for that snapshot. All results presented in this section are indicated as normalised values, i.e., as the relative difference to the reference value. Smaller differences indicate better performance. For example, assume that the best throughput (reference value) found over all experiments, concerned with a given snapshot, is 1750 lots per month. Further assume that a particular experiment (e.g., $P = 50$ and $filter = no$) indicates a throughput of 1700 lots per month. The difference (50 lots per month), relative to the reference value (1750 lots per month) is is thus 2.9%.

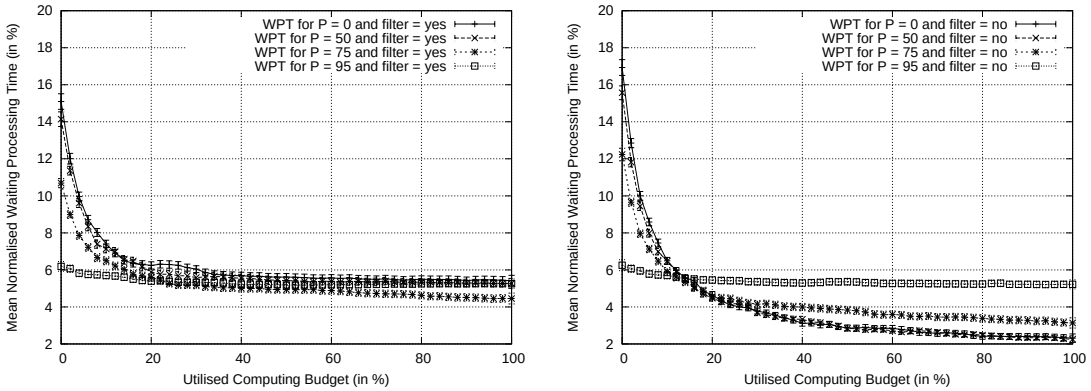
Figures 5.7 and 5.8 illustrate the convergence behaviour of the WIA process for throughput and working processing time, respectively. More specifically, each figure illustrates the mean of the normalised performance values and their 95% confidence intervals over all 35 snapshots.

The results show the impact of specialisation on the performance of the WIA process. Using higher values for P leads to increased performance (i.e., smaller normalised values) compared with the general case of $P = 0$ which considers all station families, regardless whether they are "critical" or not. However, the results also show the fading effect of the problem solver agent on the overall performance of the physical system with increasingly higher values for P. For example, for $P = 95$ (see Figure 5.8b), the convergence behaviour in terms of waiting processing time clearly shows the decreased effect. This is due to the fact that for high values of P, there are only a few stations that are being controlled by the problem solver agent. As a result, the effect of the problem solver agent is very little.



(a) (b)

Figure 5.7: Convergence behaviour of the WIA process in terms of throughput for different values of P and using a setup filter (5.7a) and without using a setup filter (5.7b).



(a) (b)

Figure 5.8: Convergence behaviour of the WIA process in terms of waiting processing time for different values of P and using a setup filter (5.8a) and without using a setup filter (5.8b).

However, since the focus of attention is always on the most critical station families, the effect on the throughput is still noticeable. This is due to the fact that the throughput can be severely affected by a few stations that are not operating efficiently.

With exception for very high values of P, the results indicate that performance of the WIA process in terms of quality of solutions (throughput and working processing time) generally increases with higher values for P. In other words, up to a certain degree, higher levels of specialisation lead to better performance. However, the results also show that specialisation is not necessarily beneficial. For all evaluated cases, the WIA process performs better if *no* setup filter is used. Although the search space is smaller if a setup filter is applied, it does not lead to better performance. The only difference between using and not using a setup filter is the possibility of implementing setups for which there is no significant demand. However, filtering is based on a 24 hour forecast and does not cover the effects in setup demand caused by actions of the problem solver agent. As explained in Section 5.3, due to the complexity of the semiconductor manufacturing process it is possible that changes to some part of the physical system will affect other parts of the physical system. For example, changing the setup for a particular station will affect the queueing situation of other station families: a station (now using a different setup) will process different wafer lots. Upon completion of processing at this station, these lots may proceed to different station families as the lots which would have been processed if the setup of the station would not have been changed. Therefore, the demand situation at different station families is changing, possibly causing significant demand of a particular setup which has been filtered out. This explains why the performance is generally better if no setup filter is applied.

5.5 Discussion

Although the problem solver agent is (in theory) capable of making decisions autonomously, it can be expected that in a real-world application scenario, the problem solver agent is not directly instructing operators but rather a senior operations manager, who takes the suggestions by the problem solving agent and (if found to be appropriate) forward them to the corresponding operators. This may lead to sub-optimal decision making because the operations manager may not immediately see the advantage of implementing changes suggested by the problem solver agent. In addition, operators do not always accurately follow their instructions. The simulation model assumes that instructions by the problem solver agent are strictly followed, i.e., human operator are assumed to always follow their instructions. Although problems with human interference are an important issue, they are beyond the scope of this work and thus not further discussed. Nevertheless for a real-world application of the problem solver agent, issues of human interference need careful consideration and have to be dealt with accordingly.

An important issue with respect to overspecialisation has been highlighted by this application. Reducing the size of the search space can improve the performance of the WIA process in terms computing budget utilisation and quality of solutions. However, when reducing the size of the search space too much or removing important solutions, the resulting search space is overly restricted and the WIA process is not capable of finding reasonably good solutions any longer. In the context of this application, the setup filter effectively reduces the size of the search space. However, it also filters out setups that might be important for improving the efficiency of the manufacturing process. Without these setups, the resulting schedules are generally inferior to schedules that have been created based on all possible setups. This has important practical implications because reducing the size of the search space makes it easier for the WIA process to find a relatively good solution (relative to other solutions in the search space). However, there

is no guarantee that a search space, after reducing it, will still contain solutions that are good enough in the context of the application. Hence, there is a trade-off between time required to find a solution and quality of solution.

The focus of this application has been a fully-automated approach: a problem solver agent observes the physical system and triggers a what-if analysis every 12 hours. Upon triggering, the problem solver agent analyses the current situation and states a problem (in terms of Γ and Ω) and solves it. A fully-automated approach, in which the problem solver agent is completely autonomous, may not be desirable in practice for various reasons. The semiconductor manufacturing process is a delicate issue and has significant impact on the profitability of the fab. A manufacturer may not be willing to leave decision making entirely up to a software system. A semi-automated approach may thus be preferred. An advantage of our approach is the availability of possible intervention points at which a user can intervene and interact with the problem solver agent. For example, instead of leaving the pre-WIA up to the problem solver agent, a user may manually specify the problem that has to be solved, i.e., the user can manually specify Γ and Ω. In this case, the intervention point would be function f_{pa}.

5.6 Summary

We have demonstrated how to use the framework to realise a problem solver agent that makes use of a class A symbiotic simulation system in the context of a semiconductor manufacturing application. The problem considered in this application is concerned with decision making regarding a schedule for changing setups of stations. We have explained how situation-specific information about the current operating conditions can be incorporated into the what-if analysis in order to change the focus of attention to problematic station families. This focus may shift any time depending on the current operating conditions. The flexibility of ECML can be used to realise a shift of focus by dynamically

adding or removing stations from the genotype model. Similarly, the precise objective of the WIA process is also dynamically changing because only waiting processing time of problematic station families is considered. However, despite the changes in the focus of the problem solver agent, the nature of the problem remains a combinatorial problem. In the next chapter, we discuss an application which (among other issues) also considers the case where the nature of the problem is changing.

Chapter 6

Application Example 2: Radiation Detection

6.1 Overview

Radiological dispersal devices (RDDs), also commonly known as "dirty bombs", are combinations of conventional explosives and radioactive material. Unlike nuclear weapons which have an immense potential for destruction, this is not the case for RDDs. Since nuclear fission does not take place, immediate destruction is caused by the conventional explosive rather than by the radioactive material. However, upon the detonation of an RDD, the radioactive material will be scattered into the environment which may have significant long-term health and economic impact on the affected population and the affected area [118]. Exposure to high levels of radioactivity can cause radiation sickness and increase the risk of cancer. Depending on the amount of radioactive material used for the attack, it might be necessary to completely evacuate and decontaminate the affected area which may take several months.

Terrorists seeking to build a dirty bomb need to procure suitable radioactive material. According to the International Atomic Energy Agency (IAEA), suitable material to build a dirty bomb can be found in almost every country [62]. The IAEA also states that "more than 100 countries may have inadequate control and monitoring programs necessary to

prevent or even detect the theft of these materials". Clearly, the best protection against the threat posed by dirty bombs is to prevent radioactive materials from falling into the hands of terrorists. However, this involves joint efforts of many countries and can only be successful in the long run, if at all. In the meantime, other solutions have to be considered to prevent dirty bomb attacks.

In the previous chapter, we have discussed a problem solver agent in the context of a semiconductor manufacturing application that has been used to solve decision making problems. In this chapter, we will describe a problem solver agent that is concerned with a model and state identification problem in the context of a radiation detection application. A hybrid class B/C symbiotic simulation system is thus employed. The symbiotic simulation system is used for classification and tracking of a moving radiation source, assuming the use of radiation detection equipment such as Geiger counters. Efficient tracking requires the problem solver agent to have a reasonably short reaction time. We present a multi-stage approach, based on ECML and MEAα, which aims at minimising the reaction time by dynamically incorporating *a priori* knowledge about the radiation source in order to improve the performance of the WIA process.

This chapter is structured as follows. In Section 6.2 we explain how radiation detection is performed. In Section 6.3 we discuss the design of the problem solver agent. In addition, we discuss how *a priori* knowledge about the radiation source can be incorporated into the genotype model to improve the performance of the WIA process. In Section 6.4 we discuss a number of experiments with focus on the impact of problem-specific knowledge on detection performance. We conclude this chapter by discussing the application example and summarise the contribution of this chapter in the context of the dissertation in Section 6.5 and Section 6.6, respectively.

6.2 Radiation Detection

There are various types of radiation such as alpha, beta, and gamma radiation. Alpha and beta radiation can be blocked by using light shielding materials such as paper or aluminium, respectively. Gamma radiation, on the other hand, requires more substantial shielding such as concrete walls or lead plates. In addition, gamma radiation is emitted by most radioactive isotopes. Therefore, detection of RDDs should focus on gamma radiation. Various types of detector devices can be used to detect gamma radiation. Geiger counters are commonly used and count the number of detected radiation particles during a certain time period. The higher the particle count, the higher the intensity of radiation. However, Geiger counters do not provide information regarding the characteristics (i.e., type of substance and radioactivity) of the radiation source. More sophisticated radiation detection systems, such as spectroscopy systems, have to be used for this purpose. More sophisticated devices can also be expected to be more expensive. For practical purposes, we therefore assume that it is desirable to use low-cost devices such as Geiger counters.

6.2.1 Radiation Model

Radiation is generated by the process of radionuclide decay. Depending on the radioactive isotope, different numbers of particles with varying energy levels are emitted during this process. For example, the isotope Cobalt-60 emits two photons with energies of 1.17 MeV and 1.33 MeV during the decay process [23]. The activity of a radioactive source is proportional to the amount of the radioactive isotope and defined by the number of nuclei decaying per unit time. The unit for activity is Becquerel (Bq) which is defined as one decay per second. The decay process can be modelled as a Poisson process [134]. Particles are emitted randomly into every direction with equal probability. Whenever the path of a particle intersects with a detector, the counter value for this detector is increased. This model reflects very well on how Geiger counters actually work.

While activity refers to the number of nuclei decay, intensity refers to the number of particles per unit time measured at some location. Both metrics are similar and intensity is therefore also measured in Becquerel. The intensity I_2 of radiation at a certain distance d_2 to the radiation source can be calculated by using the Inverse Square Law and a reference value for the intensity I_1 at another distance d_1:

$$\frac{I_1}{I_2} = \frac{d_2^2}{d_1^2} \tag{6.1}$$

This means that if the distance is doubled, the intensity is decreased by a factor of four. The measured intensity also depends on the different kind of materials a particle has to pass through before being detected. In general, the higher the density, the higher the probability that particles are absorbed. The ability of a particular material to absorb radiation is expressed by the linear attenuation coefficient μ which is measured in units of cm^{-1}. The transmitted intensity I can be calculated based on μ, the initial intensity I_0, and the distance d a particle has to travel through the material:

$$I = I_0 e^{-\mu d} \tag{6.2}$$

In this context, the half-value thickness $d_{1/2}$, measured in units of material thickness (cm), is of particular interest for practical purposes. It represents the required thickness of a certain material to reduce the intensity by 50%:

$$d_{1/2} = \frac{0.693}{\mu} \tag{6.3}$$

The linear attenuation coefficient depends on the material and the gamma photon ray energy. For example, lead has a linear attenuation coefficient of 0.5545 cm^{-1} for a photon energy of 10.0 MeV [80]. This means that a lead plate with a thickness of 1.25 cm is required in order to attenuate the intensity of a 10.0 MeV photon energy

emitting radiation source by 50%. The linear attenuation coefficient is important for modelling because different isotopes emit particles at different energy levels. This has to be considered in the modelling process. For this purpose, we use different models for each isotope.

6.2.2 Environment

The environment used in this application example is based on the lower ground floor plan of the Kalmar Museum of Art [5]. This object has been chosen because of the availability of a floor plan as well as the fact that the building structures are made of concrete. As explained above, the attenuation model is based on linear attenuation coefficients which depend on materials used in the environment and the radioactive isotope. Since materials such as concrete, lead, and steel are common shielding materials against radiation, their attenuation coefficients or the resulting half-value thickness for different isotopes are readily available. For this reason, a building that is primarily made of concrete, such as the Kalmar Museum of Art, has been chosen. In order to create a model, a floor plan of the building has been used and for simplicity it is assumed that all wall structures are made of concrete.

The dimensions of the environment are approximately 41.3 x 45.5 meters. With a resolution of 4.55 cm/pixel, the dimensions are 908 x 1000 pixels. Based on the floor plan of the museum, a virtual two-dimensional environment has been created. In addition, we have placed five static detectors into the virtual environment at the following positions: $(91, 136)$, $(685, 133)$, $(628, 891)$, $(121, 688)$, and $(432, 502)$. As for the coordination system, point $(0, 0)$ refers to the lower left corner of the environment. The resulting radiation attenuating properties of the building, as well as the locations of the radiation source and the detectors, are illustrated in Figure 6.1.

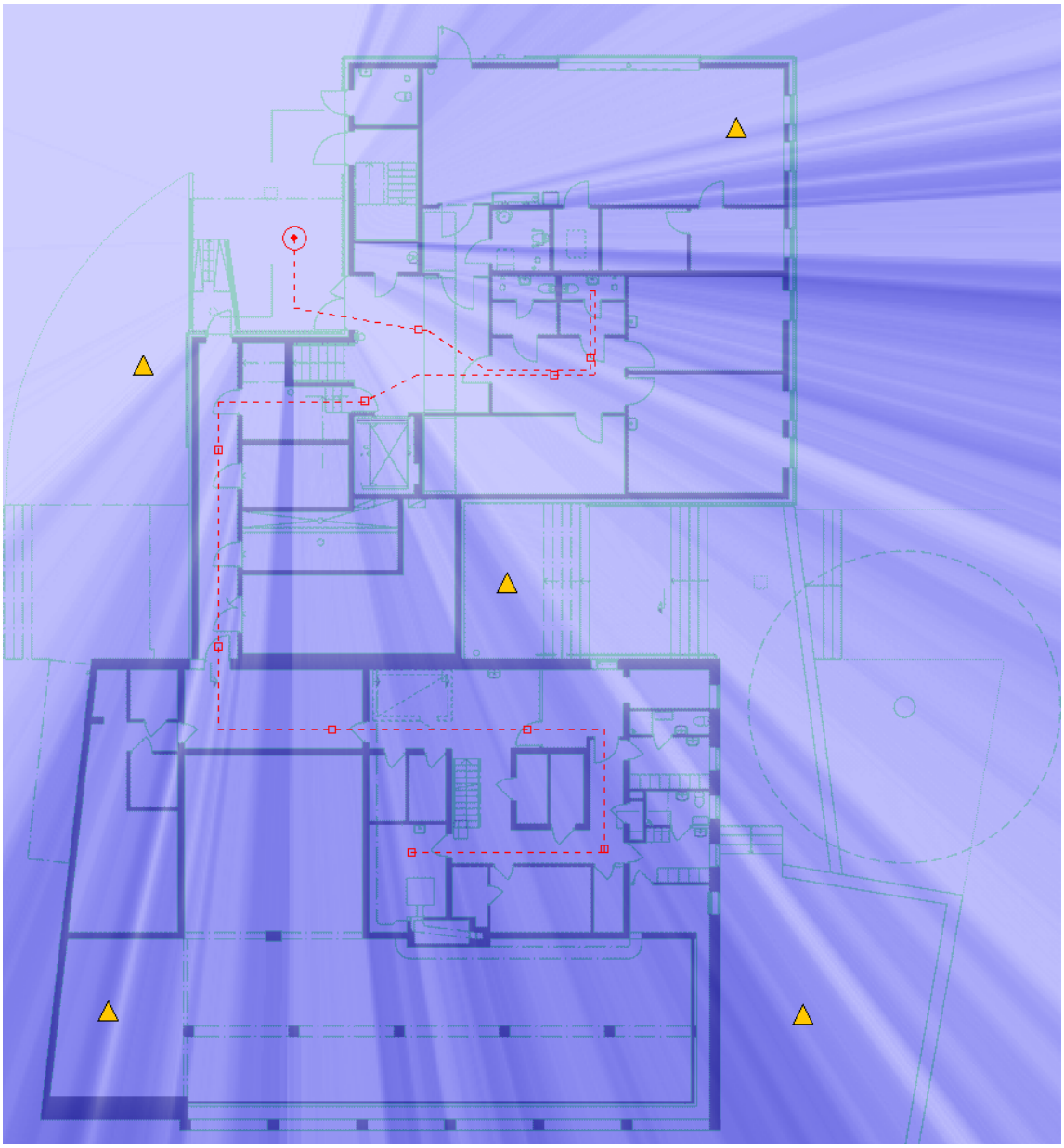


Figure 6.1: Example of radiation absorption in an environment which has been created based on a floor plan taken from [5]. Wall structures are assumed to be made of concrete. The position of the radiation source and various detectors are indicated by a circle and triangles, respectively. Higher radiation intensities are indicated by a brighter blue. The path of the radiation source is illustrated as a dashed line with rectangular markers, denoting 10%, 20%, ..., 100% of the total path length relative to the starting point.

6.2.3 Classification and Localisation

Trilateration is a method to determine the location of objects if information regarding their distance to reference points is known. Geiger counters typically do not provide this information. Taking radiation intensity as a substitute for distance will lead to unsatisfactory results because the intensity at a specific location depends on various factors such as the radiation absorbing properties of the surrounding environment. However, based on a sufficiently accurate model of the environment and measurements of radiation intensities at multiple reference locations, it is possible to compute the location of the radiation source and determine its characteristics.

Knowledge about building layouts and materials can be used to create a detailed absorption model of a particular area of interest. The radiation absorption model can be used to estimate the radiation intensity at a particular location. This also requires that all necessary information about the radiation source is available. Three factors are important in this context: i) the kind of radioactive isotope (e.g., Cobalt-60), ii) the amount of the substance (i.e., the total radioactivity), and iii) its location.

If information about radiation intensities at reference locations is available, the absorption model can be used in reverse fashion to classify and localise the radiation source. Actually measured values of radiation intensities can be used for this purpose. Solving the absorption model in reverse fashion is to seek the answer to the question regarding what radiation source, located at which location, would result in the observed radiation pattern. More formally, let the vectors $\mathbf{I^m} = (I_0^m, I_1^m, \ldots, I_{n_d-1}^m)$ and $\mathbf{I^e} = (I_0^e, I_1^e, \ldots, I_{n_d-1}^e)$ denote the measured and estimated radiation intensity of various locations, where the length n_d of these vectors is the number of detectors in the environment. The exact location of the detectors is assumed to be known and expressed by the two vectors **xd** and **yd**, where the i-th element of these vectors is denoting the position of the i-th detector in the corresponding dimension.

We consider four different radiation isotopes, each of which is represented by a corresponding model that describes the properties of each isotope:

$$\mathcal{M} = \{M_{Cs137}, M_{Co60}, M_{Ir192}, M_{Ra226}\} \tag{6.4}$$

As explained in Section 2.3.2 (see Equation 2.1), in general, a simulation model includes parameters for control (PC), external influence (PE), and simulation period (r). However, the four radiation models (M_{Cs137}, M_{Co60}, M_{Ir192}, and M_{Ra226}) considered here do not need any parameters for control or external influence: PC and PE can thus be omitted. In addition, we intend to estimate radiation intensities momentarily and not over a period of time. Therefore, r is also omitted. State information PS includes the radio activity as and the position xs and ys of the radiation source. In addition, the positions of the detectors is also included:

$$PS = \{as, xs, ys, \mathbf{xd}, \mathbf{yd}\} \tag{6.5}$$

While **xd** and **yd** are known (i.e., **xd** and **yd** are part of the known state information S' provided by the sensors), this is not the case for as, xs, and ys (part of the unknown state information S''). The state of the physical system is thus only partially known. We further distinguish between $PS'_t = \{\mathbf{xd}, \mathbf{yd}\}$ and $PS''_t = \{as, xs, ys\}$ (i.e., $PS_t = PS'_t \cup PS''_t$ analogue to $S_t = S'_t \cup S''_t$) which represent the known and unknown part of the state at time t. Given a model $M_j \in \mathcal{M}$ that reflects the radiation source and a complete set of state information at time t, the radiation intensities at the reference locations (i.e., at the locations of the detectors) at time t can be estimated as follows:

$$\mathbf{I^e}_t = M_j(PS_t) \tag{6.6}$$

In this application we assume a single radiation source located in a two-dimensional space. Radiation is generated during the decay process of the isotope and the half-life

indicates how long it takes until half the amount of material has decayed. Caesium-137, Cobalt-60, Iridium-192, and Radium-226 isotopes have half-lifes of 30.07 years, 5.27 years, 73.83 days, and 1600 years, respectively [23]. For the time periods considered in this application (minutes/hours) it is therefore reasonable to assume that the amount of radiating material remains constant, i.e., the state variable as is time independent.

The problem of identifying the correct model M_j and corresponding values for state parameters as, xs, and ys can be formulated as an optimisation problem. The objective is to find a solution for which the estimated radiation intensities $\mathbf{I^e}$ is closest to the measured radiation intensities $\mathbf{I^m}$. In other words, the objective is to minimise the error ϵ between the estimated and measured intensity values at the n_d detector locations:

$$\text{Minimise } \epsilon = \sum_{i=0}^{n_d-1} \frac{|I_i^e - I_i^m|}{I_i^e} \tag{6.7}$$

6.3 Problem Solver Agent

The purpose of the problem solver agent in this application is to classify and track a moving radiation source. As explained in the previous section, this can be done by identifying an appropriate radiation model and corresponding input parameters in order to match the simulation output with observations in the physical system. The problem is thus concerned with both, model identification and state identification. For this purpose a hybrid class B/C symbiotic simulation system can be used. In the previous chapter, we have used the framework to realise a class A symbiotic simulation system for control of semiconductor manufacturing equipment. The problem solver agent for radiation detection is realised using the same framework. However, in contrast to the previous application, this time the problem solver agent does not control the physical system. Therefore, only framework components which belong to the perception layer and the

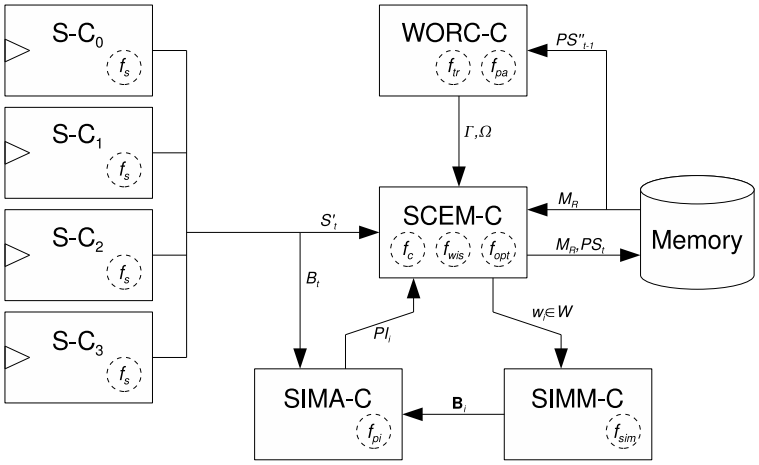


Figure 6.2: Design of the problem solver agent, based on a hybrid class B/C symbiotic simulation system, for radiation detection.

processing layer are needed. Figure 6.2 illustrates the design of the problem solver agent for the radiation detection application.

The purpose of this application is to track a moving radiation source. Tracking is a process that requires to continuously perform what-if analyses, i.e., once a what-if analysis has finished, the next one is triggered without any delay. What-if analyses are thus performed pro-actively and do not require any triggering condition. Only incomplete state information PS'_t is available at time t, i.e., the location of the detectors (**xd** and **yd**). Information regarding the radiation source (as, xs, and ys) is unknown. Estimations regarding unknown state information PS''_t is available upon completion of a WIA process. Once available, this information can be used in subsequent what-if analyses by the pre-WIA process which incorporates it as problem-specific knowledge Γ into a corresponding genotype model. By doing so, the WIA process benefits from a smaller solution space. The objective Ω for every what-if analysis is to minimise the estimation error ϵ.

A two-stage approach is applied to determine the characteristics and the location of the radiation source. Depending on the detection stage, the WIA process is concerned with either the identification of the reference model $M_R \in \mathcal{M}$ and/or the state of the

physical system PS_t at time t.

- *Stage 1 detection* is concerned with identification of the correct isotope (reference model M_R) and radioactivity (state information as) of the radiation source. No *a priori* information is used at this stage, i.e., only information regarding PS'_t is available at time t.

- *Stage 2 detection* is primarily concerned with tracking of the radiation source as it moves in the environment. This stage makes use of *a priori* information about the radiation source and consists of two parts. Previous estimations PS_{t-1} from time $t-1$ are available and can thus be used to improve the search process in the second stage. The first part (Stage 2a) is concerned with initial localisation of the radiation source and uses information about the isotope and radioactivity. The second part (Stage 2b) is responsible for continuous tracking and also uses information regarding the estimated position of the radiation source.

For each what-if scenario i, the simulation results $\mathbf{SB}_i$ provide information regarding the estimated radiation intensities (i.e., $\mathbf{I^e}$). This information is compared with sensor data B in order to calculate a corresponding PI value (i.e., ϵ) according to Equation 6.7. Earlier, in Section 2.3.2, we explained that the simulation results **SB** is a vector with a length that corresponds to the length of the simulation. In this application, the vector **SB** has a length of 1, i.e., the simulation results are only concerned with estimates of the current state estimation and not with forecasts. Upon completion of the WIA process, the best what-if scenario is decoded in order to obtain the reference model M_R and estimated complete state information PS_t.

6.3.1 Representation of Γ for Stage 1

Initially there is no *a priori* knowledge available, i.e., neither the characteristics nor the location of the radiation source is known. Therefore, all possible isotopes, radioactivity

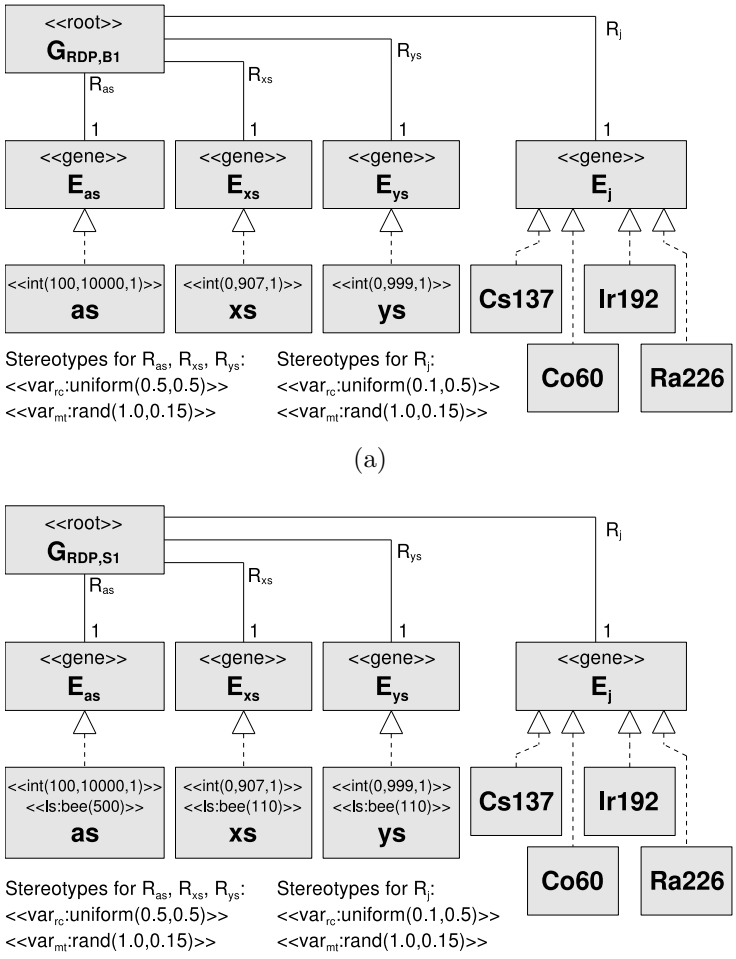


Figure 6.3: Genotype models $G_{RDP,B1}$ (6.3a), which encodes structural information only, and $G_{RDP,S1}$ (6.3b), which also encodes advanced processing information.

levels, and locations have to be considered, i.e., the entire solution space. A basic genotype model for this case is $G_{RDP,B1}$ which is illustrated in Figure 6.3a. This genotype model encodes only structural information (using genes and integer alleles). Advanced processing information is included in genotype model $G_{RDP,S1}$, illustrated in Figure 6.3b, which also includes local search regarding alleles concerned with radioactivity (as) and position (xs and ys).

The structural information provided by both genotype models is the same and imposes

a genotype structure which consists of four genes: E_j, E_{as}, E_{xs}, and E_{ys}. Four possible kinds of radioactive isotopes are considered in this example: Caesium-137, Cobalt-60, Iridium-192, and Radium-226. Each of them is represented as corresponding symbolic allele in the genotype model. Gene E_{as} encodes the radioactivity of the radiation source as an integer value ranging from 100 to 10000 GBq with a resolution of 1 GBq. The two genes E_{xs} and E_{ys} are used to encode the position of the radiation source as integer values within a range of $[0, 907]$ and $[0, 999]$, respectively, with a resolution of 1. This results in a solution space with a size of $4 \times 9901 \times 908 \times 1000 \approx 3.6 \times 10^{10}$ solutions.

In addition, genotype model $G_{RDP,S1}$ includes advanced processing information. For this purpose, a local search method is indicated. In this application example, the bee algorithm [112] is used to perform local search for as, xs, ys alleles that encode the radioactivity and the position of the radiation source. The bee algorithm needs to know the size of the neighbourhood in order to perform a local search. Stereotype `<<ls:bee(500)>>` attached to allele as indicates that a bee algorithm with a neighbourhood size of 500 GBq is used. The neighbourhood size is thus approximately 5% of the radioactivity range considered in this applications. Furthermore, stereotype `<<ls:bee(110)>>` attached to alleles xs and ys indicate that the bee algorithm is also applied to position alleles. A neighbourhood size of 110 pixels is used along both dimensions. This corresponds to an area of 5m^2.

6.3.2 Representation of Γ for Stage 2

Once the radiation substance and the amount of its radioactivity have been determined, the detection process only needs to be concerned with the location. Localisation is divided into two sub-stages: Stage 2a is concerned with the entire area and Stage 2b only considers a small search area. Since the type of radioactive substance and its radioactivity has been determined in the previous step (Stage 1), it is possible to re-use this *a priori*

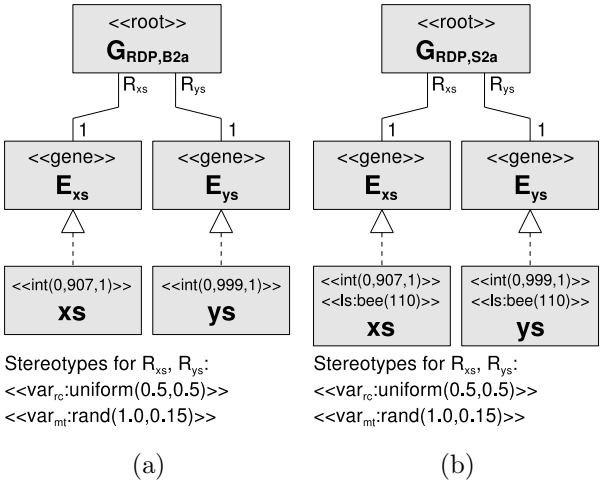
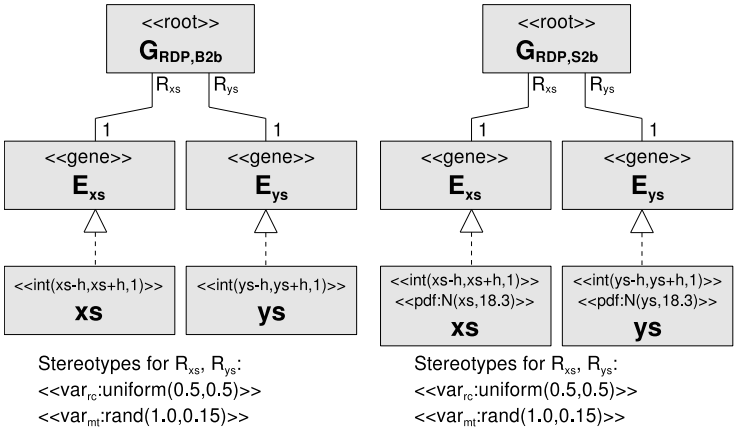


Figure 6.4: Genotype model $G_{RDP,B2a}$ (6.4a), with structural information only, and $G_{RDP,S2a}$ (6.4b), including advanced processing information, used for stage 2a detection.

information for subsequent tracking of the radiation source. Hence, the WIA process does not need to be concerned with identification of the isotope and radioactivity anymore, i.e., the isotope and the activity genes can be removed. Similar to Stage 1, we consider two genotype models: genotype model $G_{RDP,B2a}$, illustrated in Figure 6.4a, includes only structural information and genotype model $G_{RDP,S2a}$, illustrated in Figure 6.4b, which also includes advanced processing information.

As mentioned above, real-time constraints have to be considered. In order to track a moving radiation source, the problem has to be solved in a timely manner. One way of achieving this goal is to reduce the size of the solution space. In contrast to the first detection stage, using genotype models $G_{RDP,B1}$ and $G_{RDP,S1}$ and considering the entire solution space, the genotype models used in Stage 2a are only concerned with the position of the radiation source. The resulting solution space has a size of $908 \times 1000 \approx 908 \times 10^3$ solutions. Compared with the solution space for Stage 1, this is significantly smaller. Another way to improve reaction time of the problem solver agent, is to use advanced processing information to enable MEAα to work more efficiently. In this case, we use



(a) (b)

Figure 6.5: Genotype model $G_{RDP,B2b}$ (6.5a), with structural information only, and $G_{RDP,S2b}$ (6.4b), including advanced processing information, used for stage 2b detection.

local search around the previously estimated position of the radiation source.

An even more compact solution space can be obtained once the location of the radiation source has been approximated. Since the speed of the source is limited, subsequent localisation cycles only need to consider a relatively small search area. Limiting the search area can be done by restricting the value ranges for xs and ys alleles. This is illustrated in Figure 6.5.

The structural information provided in genotype models $G_{RDP,B2b}$ and $G_{RDP,S2b}$ restrict the solution space to a small area, based on *a priori* knowledge regarding the position (xs, ys) of the radiation source which becomes available after completion of Stage 2a. This information (and all subsequent position estimations) is used for all Stage 2b detection steps in order to efficiently track the target as it moves. For this purpose, a search area of $(2h)^2$ is considered. In this application we consider a search area of 5m^2, hence $h \approx 55$ (based on a resolution of 4.55 cm/pixel). The resulting size of the solution space becomes $110 \times 110 = 12100$.

For example, in genotype model $G_{RDP,S2a}$, as illustrated in Figure 6.5, the search

area is restricted to value ranges of $[xs - h, xs + h]$ and $[ys - h, ys + h]$, respectively. In addition, local search is no longer applied to alleles xs and ys. Instead, normal distribution functions are assumed using the `<<pdf:N(xs, 18.3)>>` and `<<pdf:N(ys, 18.3)>>` stereotype, respectively. This effectively biases the search to locations that are close to the last known location of the radiation source.

6.4 Experiments

6.4.1 Impact of Problem-specific Knowledge

Two kinds of problem-specific information provided in the genotype model specification can be distinguished: structural information and advanced processing information. Figures 6.3, 6.4, and 6.5 illustrate the genotype models used in different stages of the detection process, including structural and processing information. We evaluate the various kinds of problem-specific information that is reflected in the genotype model in two steps.

1. In the first step, we are only concerned with the structural information and omit any form of advanced processing information, i.e., we use basic genotype models $G_{RDP,B1}$, $G_{RDP,B2a}$, and $G_{RDP,B2b}$. These genotype models use only basic processing information regarding variation operators. This processing information is essential to the working of an EA and thus cannot be omitted. Processing information regarding local search and biased allele probabilities, on the other hand, are not essential to the working of an EA and can thus be considered optional.

2. In the second step, we evaluate how the various additions of advanced processing information (local search and biased allele probabilities) affect the performance of the WIA process, i.e., for the second step of the evaluation we use genotype models $G_{RDP,S1}$, $G_{RDP,S2a}$, and $G_{RDP,S2b}$.

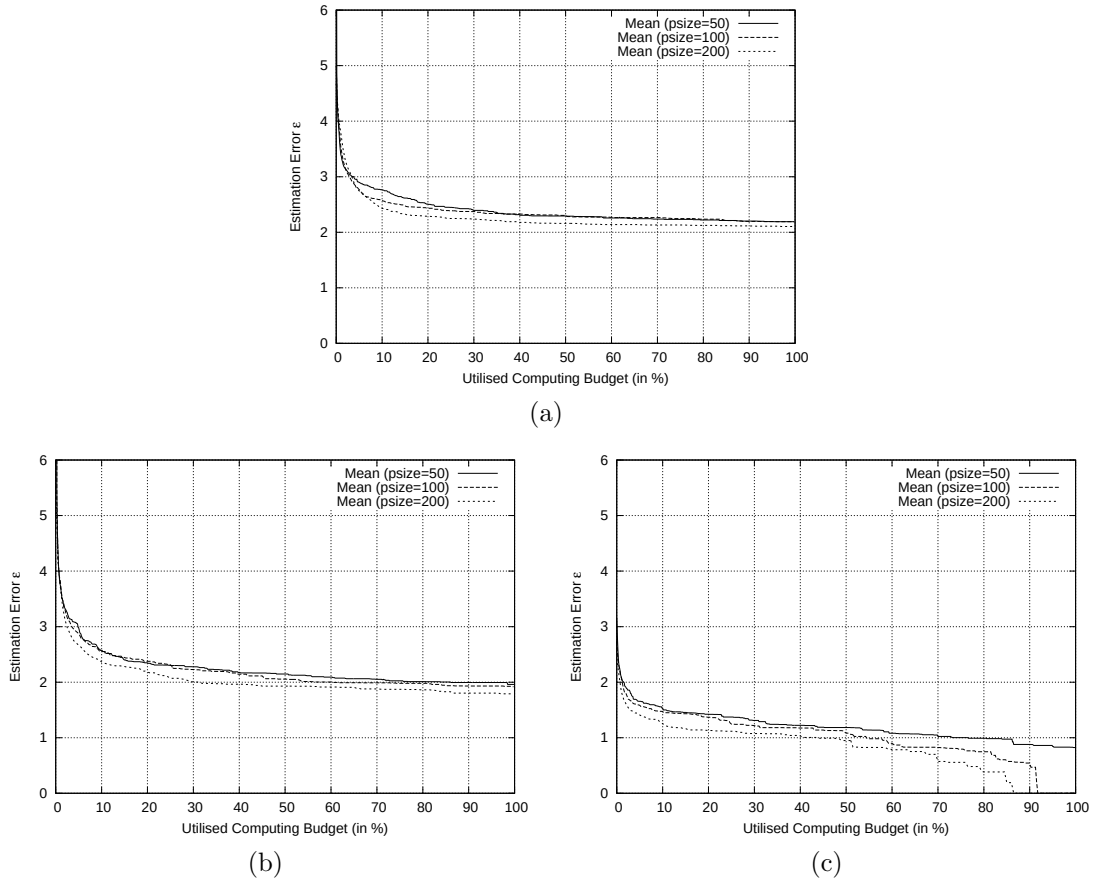


Figure 6.6: Convergence behaviour in term of ϵ for Stage 1 (6.6a), Stage 2a (6.6b), and Stage 2b (6.6c), depending on computing budget utilisation and population size ($psize = \{50, 100, 200\}$), using genotype models $G_{RDP,B1}$, $G_{RDP,B2a}$, and $G_{RDP,B2b}$, respectively.

In order to evaluate the performance of the WIA process, we consider the convergence behaviour using a fixed computing budget of 40000 simulations and three different settings for the population size ($psize = \{50, 100, 200\}$). The same computing budget is used for all experiments to ensure that different settings for population size do not gain an unfair advantage. For the basic genotype models, computing budget is equal to the product of population size and total number of generations. Figure 6.6 illustrates the convergence behaviour in terms of the estimation error ϵ for Stage 1, Stage 2a, and Stage 2b, depending on computing budget utilisation and population size.

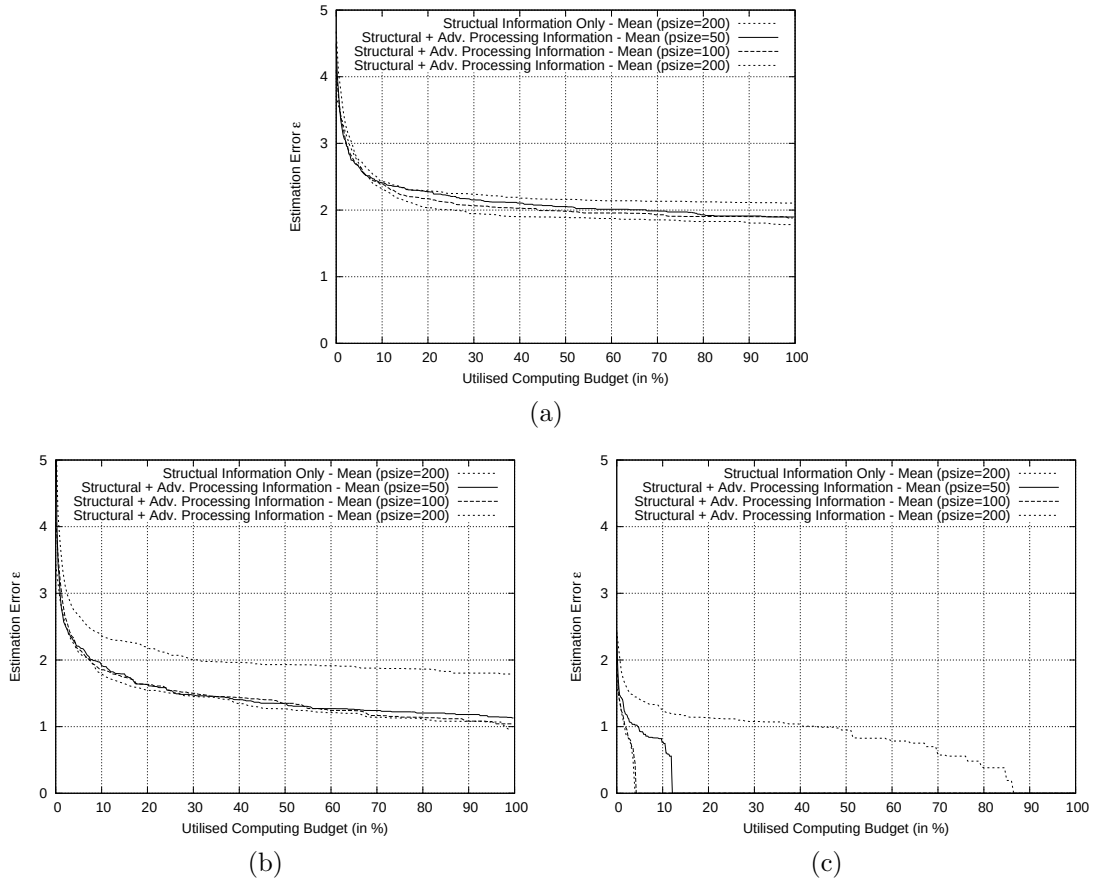


Figure 6.7: Convergence behaviour in term of ϵ for Stage 1 (6.7a), Stage 2a (6.7b), and Stage 2b (6.7c), depending on computing budget utilisation and population size, using genotype models $G_{RDP,S1}$, $G_{RDP,S2a}$, and $G_{RDP,S2b}$, respectively. For comparison, the results for basic genotype models $G_{RDP,B1}$, $G_{RDP,B2a}$, and $G_{RDP,B2b}$ ($psize = 200$) are also illustrated.

We evaluate the impact of advanced processing information using the same experimental setup as for the evaluation of structural information. A fixed computing budget is assumed and three different population sizes are considered. In addition, advanced processing information is used to further improve the search process. Figure 6.7 illustrates the convergence behaviour in terms of ϵ for Stage 1, Stage 2a, and Stage 2b when using advanced processing information.

The results confirm the assumption that smaller solution spaces lead to better performance. The estimation error is lowest for Stage 2b which is concerned with the smallest

solution space. Furthermore, the results indicate that a larger population size slightly improves the convergence behaviour, i.e., better solutions are found faster. This is true regardless whether advanced processing information is used or not. However, faster convergence can be achieved by using advanced processing information. Given the real-time requirements for this application, this is an important advantage. In general, a larger population allows a better coverage of the solution space. On the other hand, smaller populations allow to evaluate more population generations, given the same computing budget. However, as the evolutionary process is creating new generations, the diversity of the population is decreasing as offspring becomes increasingly more similar. This results in a search that is highly focused to a specific area in the solution space. Although the total computing budget is the same, smaller populations produce worse results. We attribute this to the faster decline in diversity which is more pronounced with an increasing number of generations.

In case of Stage 2b, the search converges towards the optimal solution, i.e., the WIA process identifies the correct position of the radiation source (isotope and activity are considered known from the preceding Stage 1). This is not surprising as the computing budget is larger than the solution space, i.e., the computing budget would be sufficient to exhaustively evaluate all possible solutions in the solution space. Although our approach does not perform an exhaustive search it still eventually finds the optimal solution. However, as illustrated in Figure 6.6c, for $psize = 200$ approximately 85% of the computing budget is needed in order to find the optimal solution for which $\epsilon = 0$ when using the basic genotype model. This corresponds to a total number of 34000 simulations. As explained towards the end of Section 6.3.2, the size of the solution space is 12100 solutions. The total number of simulations is thus nearly three times the amount of solutions in the solution space. This performance is worse than an exhaustive search. This is due to the fact that EAs are stochastic processes and may re-evaluate solutions that have been

evaluated before. In this case, it takes longer for the EA to find the optimal solution as compared to an exhaustive search.

The performance of the WIA process can be enhanced by specifying advanced processing information. In this application example, we make use of two kinds of advanced processing information: local search and customised allele probability distribution. Local search is applied in Stage 1 and Stage 2a to further improve the accuracy of estimating radioactivity and position of the radiation source, i.e., a local search method is applied to as, xs, and ys alleles. A customised allele probability distribution function is used in Stage 2b to bias the search to a position which is close to the last known position of the radiation source. This is based on the assumption that the radiation source cannot arbitrarily "jump" from one position to another. The experimental results, illustrated in Figure 6.7, show the effects of using advanced processing information. In Stage 1, using local search improves the WIA process only slightly (when comparing the results for a population size of 200). The advantage of using advanced processing information becomes more obvious when considering improvements in Stage 2a and, in particular, Stage 2b. The optimal solution can be identified using approximately 5% (for population sizes of 100 and 200) of the computing budget as compared to approximately 85% when advanced processing information is not used.

6.4.2 Quality of Detection

For dynamic radiation detection of a moving target, initial classification (i.e., Stage 1) is crucial because successful tracking depends on reliable knowledge regarding the characteristics of the radiation source. As mentioned above, the characteristics of a radiation source refer to the type of radiating material (isotope) and the amount of radiation material (radioactivity). Failing to estimate the characteristics will affect the reliability of subsequent tracking. With respect to estimation of the type of material,

Table 6.1: Success rate (in %) for correct classification of the isotope depending on the number of WIA process replications (R). For example, a success rate of 20% indicates that in 20% of experiments, the corresponding isotope has been successfully identified.

R	Caesium137	Cobalt60	Iridium192	Radium226	All
1	56.0	48.0	86.0	52.0	60.5
2	68.0	47.0	89.0	62.0	66.5
3	67.0	55.0	92.0	62.0	69.0
4	81.0	57.0	90.0	67.0	73.8
5	76.0	65.0	90.0	62.0	73.3
Avg	69.6	54.4	89.4	61.0	
Min	56.0	47.0	86.0	52.0	
Max	81.0	65.0	92.0	67.0	

evaluation is simple: either the isotope has been correctly identified or not. As for radioactivity as, some error in the estimation is not going to affect the detection quality significantly. The experiments in this section are concerned with the effect of estimation errors on the quality of detection.

As explained above, the WIA process is a stochastic process and multiple replications may improve the quality of solutions. In the context of this application, this is an important issue for identification as subsequent tracking will incorporate knowledge regarding the characteristics of the radiation source obtained during the Stage 1. For this reason we analyse the effects of performing multiple WIA process replications on classification. For each isotope, there are 100 experiments, each of which uses a random value for radioactivity and position of the radiation source. As explained above, different solutions are evaluated in terms of ϵ. In case of multiple WIA process replications, the best solution found over all replications is considered. Table 6.1 shows the success rate (in %) for correct detection of the isotope.

In general, the experimental results confirm the assumption that the solution quality improves with the number of replications. This is the case regardless of the actual isotope used for the radiation source. However, the results also suggest that some isotopes are

more difficult to identify than others. For example, the success rates for Iridium-192 are between 86.0% and 92.0%. This is in contrast to Radium-226, for instance, for which the success rate is significantly lower between 52.0% and 67.0%. This difference is due to the fact that different isotopes cause different radiation attenuation patterns as a result of the different linear attenuation coefficients that are characteristic for each isotope. The WIA process may be misguided in case measured radiation intensity patterns are very similar to multiple estimated radiation intensities patterns (based on different what-if scenarios).

The primary objective of the detection system is to localise a radiation source and continuously track it, as it moves within the environment. For successful tracking, initial localisation is important because all subsequent tracking cycles will be limited to a small search area. For this purpose, experiments are performed concerned with Stage 2a. For each experiment, a radiation source is placed randomly into the environment using a specific isotope. We consider the initial localisation accuracy of the system under various conditions regarding the outcome of Stage 1 detection. A condition is the combination of correct/incorrect identification of the isotope and an activity estimation error ranging from 0% to 49% in steps of 1%. For each condition, a total of 400 experiments is performed (100 for each isotope). The results are illustrated in Figure 6.8 and show the localisation accuracy in terms of position estimation error (i.e., the distance between the actual and estimated position) with respect to the activity estimation error (i.e., the error between the actual and estimated radioactivity) and correct/incorrect isotope identification.

The experimental results confirm our assumption that the localisation accuracy is significantly lower if the isotope has not been classified correctly. Furthermore, the results indicate that the localisation accuracy is declining with increasing activity estimation error. Initial localisation is followed by continuous tracking as the radiation source is moving. In order to evaluate the tracking ability we perform a number of experiments

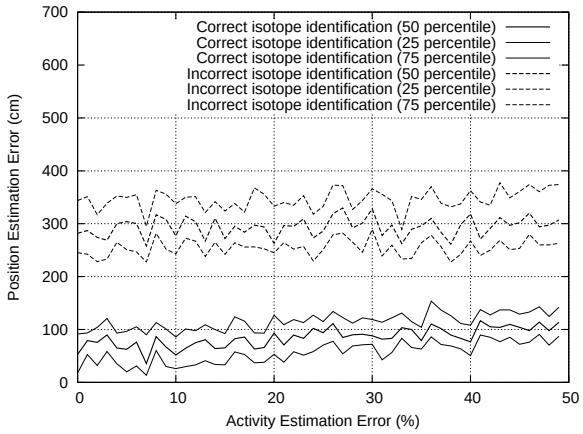


Figure 6.8: Effect of estimation errors in stage 1 on initial localisation accuracy in stage 2a.

with a moving radiation source. For these experiments it is assumed that Stage 1 has successfully identified the isotope and determined the radioactivity with an accuracy ranging between 0% and 20% (in steps of 5%). Furthermore, we assume that every Stage 2b detection cycle is completed within one second. As for the speed of the radiation source, we consider values ranging between 2 km/h (slow walking) to 8 km/h (running). For each time step we evaluate the error between the estimated position and the actual position of the radiation source. We consider tracking successful if the estimated position is within one meter of the actual position. A total of 400 experiments is performed (100 for each isotope). The results are illustrated in Figure 6.9 and show the success rate for tracking.

In general, it is easier to track a radiation source if it is moving slower. At faster paces, there is an increasing probability that the problem solver agent is losing the radiation source. Once lost (i.e., once the radiation source has moved beyond the small search area considered in Stage 2b), it is difficult for the problem solver agent to continue tracking the location of the radiation source. During the experiments, the size of the search area has been kept constant. With increasing speed, the problem solver agent is more likely to lose track of the moving radiation source. This is also supported by the experimental

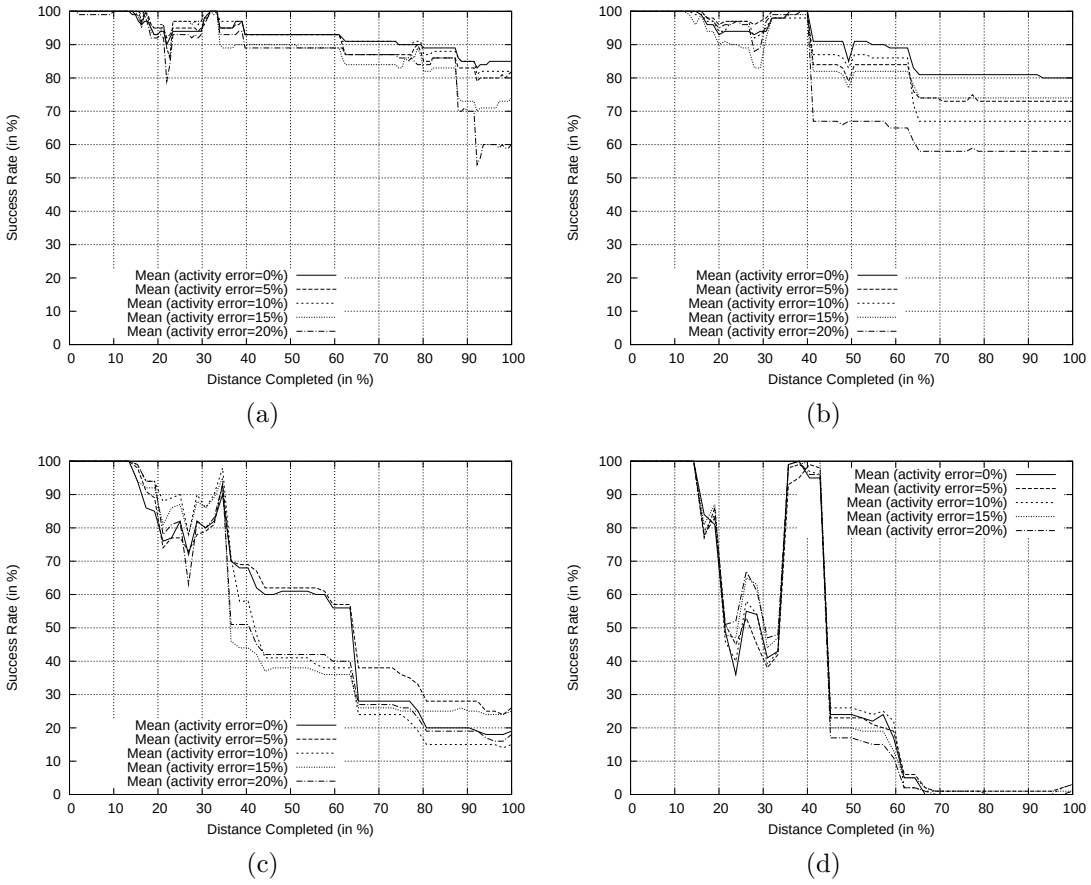


Figure 6.9: Effect of estimation errors on tracking accuracy for targets moving at 2 km/h (6.9a), 4 km/h (6.9b), 6 km/h (6.9c), and 8 km/h (6.9d) in terms of mean success rate.

results. The rate of successful tracking is rapidly decreasing at a speed of 8 km/h. However, if the radiation source happens to move back into the search area, tracking can continue. This explains the recovery at 30–40% of the completed distance (see Figure 6.9d). At higher speeds, the problem solver agent loses track of the target when it is has completed approximately 20–30% of the completed distance. The target moves towards the bathrooms and then returns the same way (see Figure 6.1). When this happens, the success rate for tracking is briefly recovering as the target moves back into the search area.

6.5 Discussion

An important issue in radiation detection is background radiation and the signal to noise ratio [142]. As explained in [17], this problem is exacerbated by shielding materials in the environment, non-uniform background radiation across a region, and the fact that a radiation source can be expected to move. Another important issue for radiation detection is the deployment of sensor networks which involves decisions regarding the type of sensors used, their quantity and placement within the environment. For example, radiation detection sensor networks have been discussed in [14, 79]. Unlike related work, which is concerned with background radiation and issues related to sensor placement, we focus on radiation detection based on an accurate radiation attenuation model of the environment. In particular, we present a method that (in theory) is not only capable of locating and tracking a moving radiation source but also identification of the radiation material using simple Geiger counter devices only.

Another important issue is the reactivity of the problem solver agent, i.e., the time required for the problem solver agent to complete a what-if analysis cycle. We describe the radiation detection problem as an optimisation problem which aims at minimising the error between the measured and estimated radiation intensities at various reference

locations. Improving the reactivity of a problem solver agent thus requires to reduce the required computing time needed to perform a WIA process. We discuss two methods that can be used for this purpose. The first method directly aims at reducing the size of the search space in the various stages. The second method aims at biasing the search towards a certain region in the search space. Both methods depend on availability of *a priori* information regarding the various characteristics of the radiation source. Experiments have shown that successful identification of the radiation source is very important for all succeeding detection cycles.

Although using *a priori* information can improve the computing time required to complete a detection cycle, this also represents a weakness of our approach. Success of subsequent detection cycles depend on correct information provided by preceding detection cycles. For example, if the isotope has been incorrectly classified, the quality of detection is severely affected. Therefore, sophisticated error handling methods need to be employed. This may include periodic re-classification of the radiation source. For example, experimental results have shown that increasing the number of replications of Stage 1 detection cycles can increase the success rate. In this context, an important problem is the discovery of an erroneous detection result. Error handling has not been covered here and is thus subject to future work.

Detection quality highly depends on an accurate model of the environment. A basic model of the environment can be easily created by using floor plans and blue prints of a building. In addition, building developers should be able to provide detailed information regarding the various materials used to construct the building. All this information can be incorporated into a detailed radiation attenuation model. In addition, including more detailed physical modelling (e.g., by considering compton scattering [22]) can be expected to improve the fidelity of the model and thus ultimately also quality of detection. More realistic models also increase the computational burden, possibly questioning the

technical feasibility of the approach. For this showcase, all simulations have been executed on general-purpose hardware (standard CPU). However, it should be possible to use special-purpose hardware (graphics processing units (GPU)s). This can be expected to increase performance significantly because GPUs were originally developed to handle the increasing demand from the gaming industry. Indeed, rendering three-dimensional environments is very similar to solving the radiation detection problem as both include calculating propagation of radiation through an environment.

6.6 Summary

We have demonstrated how to use the framework to realise a problem solver agent that makes use of a hybrid class B/C symbiotic simulation system in the context of a radiation detection application. The problem considered in this application is two-fold. First, a model identification problem needs to be solved. This problem is concerned with identification of the correct isotope model. Second, once the isotope model has been correctly identified, a state identification problem has to be repeatedly solved in order to track the movement of a radiation source. Depending on the detection stage, different degrees of knowledge about the radiation source can be incorporated into the WIA process process by reflecting available information about the radiation source in the genotype model. This application example shows how knowledge can be incorporated dynamically by using ECML. In addition, it shows the overall flexibility of the problem solving approach with ECML and MEAα because, initially, a mixed combinatorial/continuous optimisation problem and then a continuous-only optimisation problem has to be solved. Furthermore, the application shows how various forms of advanced processing information can be used at different stages to improve the performance of the WIA process.

Chapter 7

Conclusions

7.1 Summary and Contributions

Symbiotic simulation, and related paradigms such as that of DDDAS, are becoming increasingly important and interesting for researchers as well as practitioners. However, despite various efforts, in particular those in the context of DDDAS, symbiotic simulation is still in its infancy. There are a number of open research problems that need to be addressed. This dissertation has been concerned with one particular problem, i.e., the problem of dynamically changing operational conditions and the resulting challenges for the problem solving mechanism. In general, this dissertation takes a holistic approach towards a framework for symbiotic simulation. From a theoretical point of view, this dissertation has established a theory on symbiotic simulation, most notably a taxonomy of different classes of symbiotic simulation systems. In addition, from a practical point of view, we have proposed an agent-based framework that enables users to build problem solver agents that are tailored to a particular application.

Simulation-based what-if analysis is an important tool for solving real-world problems. In order to achieve the required fidelity and response time for short-term problem solving (e.g., operational decision making), the WIA process has to be performed on-line (i.e., sensor data have to be incorporated into the simulation) and in real-time (i.e., solutions have to be found within a limited period of time), respectively. Since what-if analysis

is a difficult task to perform manually, automation is required. A what-if analysis is always performed with certain objectives in mind and, from a technical perspective, is concerned with solving a particular optimisation problem regarding the physical system. Hence, automation involves the replacement of a human problem solver by an artificial problem solver agent which is capable of performing the same tasks as the human problem solver. The problem solver agent has to be capable of analysing the current situation of the physical system (using sensor data), specifying a situation-specific problem (in terms of Γ and Ω), solving the problem, and (if applicable) implementing changes in the physical system.

In the context of symbiotic simulation, dynamic specialisation of the WIA process is an important issue because operational conditions of the physical system, and thus the nature of the actual problem, may change over time. In Section 1.3 we have explained that general-purpose algorithms are typically outperformed by algorithms that have been specifically designed to solve a particular problem. For symbiotic simulation this means that, ideally, different problems that are encountered by the problem solver agent are solved by algorithms that have been specifically designed for it. However, since problem solving has to be performed within a limited period of time, there is not enough time to design a new algorithm. Instead, problem-specific information has to be dynamically incorporated into the WIA process by a suitable algorithm. To summarise, this algorithm has to be able to adapt to different situations by dynamically incorporating problem-specific knowledge in order to find sufficiently good solutions within a limited period of time.

In order to address this issue, we have proposed a flexible approach which is based on the separation of problem-specific knowledge from the algorithm implementation. As a result, we achieve the flexibility that is required in the context of symbiotic simulation. At the core of our approach is a flexible problem solver based on a descriptive language

(ECML, see Section 3.3) in combination with a meta algorithm (MEAα, see Section 3.4) that is capable of interpreting the language. The approach aims at effectively decoupling the representation of problem-specific information (that may change over time) from the algorithm implementation. By doing so, we achieve the required degree of flexibility: the same algorithm can be used to solve different kinds of problems when provided with corresponding problem-specific information described in the ECML language. Existing methods include much of the problem-specific information into the algorithm during design time. In contrast, our approach allows ad-hoc specialisation as problem-specific information becomes available during runtime.

We have demonstrated the flexibility of our approach in two applications concerned with semiconductor manufacturing (see Chapter 5) and radiation detection (see Chapter 6). In the first application, the nature of the problem is a combinatorial problem with the objective to identify a reasonably good combination of schedules for setup changes in order to improve the performance of the manufacturing process. Given the large number of stations and possible setup schedules, it makes sense to limit the attention of the problem solver agent by focusing on a number of problematic station families. The set of problematic station families is changing over time and thus the focus of the problem solver agent. In the second application, changes not only the degree of available information about a moving radiation source but also the nature of the problem itself. Initially the problem is a mixed combinatorial/continuous problem concerned with the correct identification of the characteristics of the radiation source. Once the characteristics have been identified, the problem becomes a simpler continuous problem that aims at tracking the radiation source as it moves within the environment. The same algorithm (MEAα) has been used for all these problems.

This dissertation has made the following contributions to the field of computer science. These contributions are of particular interest to the modelling and simulation community

as well as the evolutionary computing community. The contributions below are listed according to the order of their corresponding chapters.

- *Theory on Symbiotic Simulation.* We have established a theory on symbiotic simulation from a holistic perspective. The original definition of symbiotic simulation only considers closed-loop symbiotic simulation systems that are concerned with decision making problems. In addition, symbiotic simulation is significantly overlapping with other concepts such as DDDAS and on-line simulation. We have analysed the various concepts and existing applications and clearly defined symbiotic simulation in the context of its related paradigms. In particular, we focus on the WIA process which is an essential part of any symbiotic simulation system. As a result, we have defined a number of different classes of symbiotic simulation systems. Three classes (A,B,C) are concerned with optimisation. These classes have been the focus of this dissertation.

- *Flexible Problem Solver.* We have proposed a flexible problem solver method based on a descriptive language (ECML) in combination with a meta algorithm (MEAα), that allows separation of problem-specific information from the implementation of an optimisation algorithm. In our case, we have used a language based on UML and an algorithm based on evolutionary computing methods. However, the principles are generally applicable and is not limited to evolutionary computing and UML-based languages. A major difference between our contribution and the existing work is the general applicability of the language, i.e., ECML is not biased towards a particular kind of problem or application.

- *Agent-based Framework.* We have proposed an agent-based symbiotic simulation framework for constructing application-specific problem solver agents. The framework has been designed with applicability and extensibility in mind. It allows the

construction of problem solver agents that are tailored to the needs of a particular application. The framework is thus of interest to practitioners who want to use symbiotic simulation in their applications. This includes researchers and engineers, who are working in the field of symbiotic simulation.

7.2 Discussion

In this section, we discuss a number of general issues and lessons that have been learned from applying our approach to various showcases.

7.2.1 Objectives for What-if Analysis

An important issue that has to be carefully considered when applying our approach is the choice of objectives for the WIA process. This also involves consideration of appropriate way to quantify the corresponding performance measure that are of interest for optimisation. For example, in Section 5.3.2 we have explained why a simple measure for throughput has a serious draw back. In order to avoid this problem, a different approach for quantifying throughput is used (i.e., using the average throughput across all processing stages instead of the simple throughput). Furthermore, real-world problems are often multi-objective optimisation problems. In addition, many objectives are often conflicting with each other. Thus, focusing on only one objective will inevitably lead to substandard performance with respect to at least one other objective. In the semiconductor manufacturing application we have demonstrated how the problem solver agent can dynamically adapt to the requirements of a particular situation: depending on the situation, the problem solver agent is focusing on only a selection of problematic stations. Although the objectives in our application are always concerned with the performance in terms of throughput, the approach as such is generally not limited to a particular objective. In

fact, depending on the situation, the problem solver agent may chose a completely different objective (expressed by Ω) every time a what-if analysis is conducted. For example, in some applications it may make sense to temporarily focus on a particular objective while purposely ignoring other ones. This can be done by implementing a corresponding logic in function f_{pa}.

In the context of stochastic simulation, which requires the execution of multiple simulation replications, a performance measure may be indicated by a mean value and a standard deviation. Depending on the variance of the simulation results, there might be situations where it is necessary to decide between solutions with better mean performance but, at the same time, high variance and solutions with lower mean performance but also lower variance. Although lower variance may be desirable in many cases, this is not necessarily true for all cases. Variance can be handled in various ways. For example, one approach could be to explicitly consider variance as part of the optimisation objective, i.e., an objective would be to minimise the variance. Another approach would be to impose certain constraints on the variance, i.e., the objective would be to minimise/maximise a certain performance metric while maintaining a variance which is below a certain threshold. Depending on the application and current situation of the physical system, corresponding objectives and constraints can be specified as Ω independently for each what-if analysis.

7.2.2 Simulation Engine

The applicability of our approach – or rather any symbiotic simulation approach – depends on the availability of an appropriate simulation engine. Depending on the application, this could be a specially developed simulation engine or a COTS simulation package. For example, the experiments for the semiconductor manufacturing showcase discussed in Chapter 5 were originally performed using AutoSched as it is a well-known and, more

importantly, well accepted simulation engine in the semiconductor manufacturing industry. In addition to the two showcases discussed in this dissertation, our problem solver agent approach has been used in other applications as well. One particular application in the context of a reverse logistics network application uses Arena, another COTS simulation package. This application has been developed in parallel to the semiconductor manufacturing showcase as part of a project under the integrated manufacturing and service systems (IMSS) initiative Singapore [82]. Although the reverse logistic network application has not been included in the dissertation (as it would not have contributed with additional information) similar experiences were made with respect to the usability of COTS simulation packages.

Despite the advantages of COTS simulation engines, such as the wide-spread acceptance among domain experts in the corresponding industry, they have certain shortcomings when used in a symbiotic simulation application. Using data-driven paradigms, such as symbiotic simulation, that make intensive use of on-line simulations is a relatively recent trend and at the time of writing, the above-mentioned COTS simulation engines are still designed with a traditional usage pattern (single-scenario, off-line simulation) in mind. However, symbiotic simulation involves the simulation of a potentially large number of what-if scenarios during a single what-if analysis. For example, as explained in Section 5.4, what-if analyses in the semiconductor manufacturing application use a computing budget of 48,000 simulations. In addition, it is necessary to initialise the simulation runs with a specific state of the physical system. In order to efficiently simulate thousands of on-line simulations it is required that the used simulation engine has been designed with such a usage pattern in mind.

From our experience with the above-mentioned COTS simulation engines, there are two specific issues that make it difficult to use COTS simulation engines for symbiotic simulation applications. The first issue is concerned with the application programming

interface (API). In order to initialise simulation runs with the current state of the physical system it must be possible to access all internal simulation parameters and objects in order to manipulate them accordingly. For example, AutoSched provides a well documented and powerful API that allows access to many internal data objects in order to manipulate them. Arena, on the other hand, provides only very limited access and requires to develop a number of workarounds. The second issue is concerned with the support for efficient multi-scenario simulation by avoiding heavy overheads when executing a simulation. For example, both COTS simulation engines required a considerable overhead when executing a simulation due to various I/O operations. This overhead was due to activities such as connecting to license servers and reading from input files. Although these activities take only a few hundred milliseconds, the overhead is a significant performance issue. This is partially because of the large number of simulations but also due the fact that simulations cover only a very short time period. The resulting proportion of the overhead compared to the total execution time becomes significant.

7.2.3 ECML and MEA

In the two showcases, discussed in Chapter 5 and Chapter 6, we have demonstrated how dynamically incorporating problem-specific information can improve the ability of the problem solver agent to solve problems. We also explained how problem-specific information is obtained and represented by Γ. In a sense, specialisation can be considered as a way of fine-tuning parameters of the MEA. Although using the various advanced features of ECML can help to improve the efficiency and effectiveness of the problem solving process, it is not required. The bare minimum, in order to get ECML/MEAα working, requires to provide structural information (i.e., genes, alleles, and relationships). These typically follow from the application-context and the given optimisation problem. It is also necessary to provide information regarding the basic processing of genotypes

(i.e., variation operators). In addition, it is necessary to specific the population size and some termination criteria. With respect to all these, our approach does not differ from other evolutionary computing approaches which require, in most cases, the same kind of information to be specified.

There is some overhead necessary to specify a genotype model in ECML. For simple optimisation problems that do not need complex genotype structures or specialisation, it may be easier (from a users perspective) to use a simple optimisation method. For example, consider the Knapsack problems discussed in Section 3.5.1. Although the genotype model for this type of problem does not require much information to specify, it would still be easier to use a simple genetic algorithm that works with binary string genotypes and some hard-coded standard variation operators. However, in case of complex optimisation problems, a more sophisticated optimisation method may be required. In addition, depending on the nature of the problem, there might be the need for dynamic specialisation. ECML allows to specify problem-specific knowledge on a higher level without having to manipulate low-level source code of the optimisation method. This is particularly useful in the case of dynamic problems where specialisation has to be performed during runtime. Therefore, for complex and/or dynamic problems, the specification overhead is outweighed by the various advantages of the ECML approach.

Unlike existing approaches, our approach allows to (optionally) specify additional information that can be exploited by a suitable MEA in order to improve the problem solving process. One feature that makes our approach quite different from the existing approaches, is the ability to define genotype structures with arbitrary degrees of complexity and different topologies. This, in combination with a corresponding decoding mechanism, can have a significant impact on the problem solving process. For example, in Section 3.5.2 we have demonstrated how a more complex genotype structure can help to improve the outcome of the problem solving process. A crucial issue for our approach

to work is the ability to analyse the current operating conditions of the physical system in order to identify the problem and providing sufficient problem-specific information (expressed as Γ to guide the problem solving process. For example, in the semiconductor manufacturing application, we have described a method that limits the search space to problematic stations (i.e., stations which belong to station families with high waiting processing times). However, while restricting the search space may improve the performance of the problem solver, overly restricting may lead to decreased performance (as explained in Section 5.5).

7.3 Future Research

In Chapter 2 we have established a theory on symbiotic simulation. As a result, we have identified five different classes of symbiotic simulations (see Section 2.4). However, only three of them, i.e., class A – C symbiotic simulation systems, have been discussed in greater detail in this dissertation. Our taxonomy on symbiotic simulation systems also identifies other classes of symbiotic simulation systems that are not concerned with optimisation as part of their WIA process. These kind of symbiotic simulation systems deserve further research to explore possible applications, identify problems that are inherent to these kind of systems, and develop solutions for them. In this context, the control feedback of the problem solver agent to the physical system also deserves further research. We have only considered control feedback for class A symbiotic simulation systems. However, other classes of symbiotic simulation systems may also benefit from a control feedback. For example, we have explicitly distinguished between an active and a passive class B symbiotic simulation system. The class B symbiotic simulation system, discussed in Section 6.3 is a passive one. Future research should also investigate active class B symbiotic simulation systems.

In Section 2.3.2 we have explained that a what-if scenario is made up of a number of components (see Equation 2.1): a model M, an initial state PS, control parameters PC, and parameters PE that denote external influence. When modelling real-world physical systems, there are typically various sources of uncertainty (reflected by PE). If the degree of uncertainty is significant, the reliability of the what-if analysis is affected. An important research issue, which has not been discussed in this dissertation, is robustness of the what-if analysis in the context of uncertainty. Further research has to address this issue and develop methods that can cope with varying degrees of uncertainty. This not only refers to uncertainty due to lack of knowledge about the physical system during design time of the model but also uncertainty due to lack of accurate sensor data during runtime.

There are a number of limiting factors to symbiotic simulation. Limiting factors such as computing requirements can be solved by using a more efficient optimisation algorithm, for example. In addition, simulations can be performed in parallel. It is even possible to use specialised hardware, such as GPUs, is possible in case of highly specialised applications. However, the single-most limiting factor in applicability of symbiotic simulation is the model. Real-world physical systems are continuously changing. As a consequence of this, the model needs frequent updating. This is an important issue because in order to be able to rely on solutions found by the problem solver agent, it is crucial that the simulation model reflects the physical system with sufficient accuracy. Although model identification has been discussed in the context of class B symbiotic simulation systems, this issue has not been the focus of this dissertation. Further research is thus required.

An important issue in the context of stochastic simulation models, is the required number of simulation replications. This kind of replications are necessary in order to be able to aggregate statistically meaningful results. In addition, when using stochastic algorithms (such as $MEA\alpha$), then it may be necessary to perform a number of WIA

process replications as well. In other words, the same WIA process has to be repeated multiple times in addition to simulation replications. This results in a very high demand for computing power. More research is needed to identify possible relationships between the need for simulation replications and WIA process replications. Although here we have been using both, simulation replications and WIA process replications, this issue has not been adequately addressed. In this context, it should also be investigated whether simulation replications are necessary in the first place. Consider the following argument in the context of a decision making problem. If a simulation model is stochastic and if there is a very high variance in the simulation results, then the question is whether or not the model is reliable enough for decision making. One may thus argue that, given so much variance, symbiotic simulation cannot (or should not) be applied. Obviously, this argument is overly simplistic. However, the issue of replications deserves a thorough investigation and should thus be subject to future research.

7.4 List of Publications

The following papers have been accepted and published from this research:

- **H. Aydt**, W. Cai, S. J. Turner, B.P. Gan, "Symbiotic Simulation for Optimisation of Tool Operations in Semiconductor Manufacturing", in Proceedings of the Winter Simulation Conference. 2011.
- **H. Aydt**, S. J. Turner, W. Cai, Y. H. Low, Y. S. Ong and R. Ayani, "Towards an Evolutionary Computing Modeling Language (ECML)", IEEE Transactions on Evolutionary Computation, Accepted September 2010 and In Press.
- **H. Aydt**, S. J. Turner, W. Cai, and Y. H. Low, "Research Issues in Symbiotic Simulation", in Proceedings of the Winter Simulation Conference. 2009, pp. 1213–1222.

- **H. Aydt**, S. J. Turner, W. Cai, Y. H. Low, and R. Ayani, "Symbiotic Simulation Model Validation for Radiation Detection Applications", in Proceedings of the 23rd Workshop on Principles of Advanced and Distributed Simulation. 2009, pp. 11–18.

- **H. Aydt**, S. J. Turner, W. Cai, and Y. H. Low, "An Agent-Based Generic Framework for Symbiotic Simulation Systems", Multi-Agent Systems: Simulation and Applications, Taylor & Francis. 2009.

- **H. Aydt**, S. J. Turner, W. Cai, Y. H. Low, P. Lendermann, B. P. Gan, and R. Ayani, "Preventive What-if Analysis in Symbiotic Simulation", in Proceedings of the Winter Simulation Conference. 2008, pp. 750–758.

- **H. Aydt**, S. J. Turner, W. Cai, Y. H. Low, P. Lendermann, and B. P. Gan, "Symbiotic Simulation Control in Semiconductor Manufacturing", in Proceedings of the International Conference on Computational Science. 2008, pp. 26–35.

- **H. Aydt**, S. J. Turner, W. Cai, and Y. H. Low, "Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology", in Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation. 2008, pp. 109–116.

Bibliography

[1] Abhishek Agarwal and Maria Hybinette. Merging parallel simulation programs. In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS)*, 2005.

[2] Elif Akçali, Reha Uzsoy, David G. Hiscock, Anne L. Moser, and Timothy J. Teyner. Alternative loading and dispatching policies for furnace operations in semiconductor manufacturing: a comparison by simulation. In *Proceedings of the Winter Simulation Conference*, pages 1428–1435, 2000.

[3] Volkan Akcelik, Jacobo Bielak, George Biros, Ioannis Epanomeritakis, Omar Ghattas, Loukas F. Kallivokas, and Eui Joong Kim. A framework for online inversion-based 3D site characterization. In *Proceedings of the International Conference on Computational Science*, pages 717–724, 2004.

[4] S.M. Annan and J. Banks. Design of a knowledge-based on-line simulation system to control a manufacturing shop floor. *IIE Transactions*, 24(3):72–83, 1992.

[5] Tham & Videgård Hansson Arkitekter. Kalmar museum of art. archdaily.com, February 2009.

[6] Heiko Aydt, Stephen John Turner, Wentong Cai, Malcolm Yoke Hean Low, Peter Lendermann, and Boon Ping Gan. Symbiotic simulation control in semiconductor manufacturing. In *Proceedings of the International Conference on Computational Science*, volume 5103, pages 26–35, Krakow, Poland, 2008.

[7] Heiko Aydt, Stephen John Turner, Wentong Cai, Malcolm Yoke Hean Low, Peter Lendermann, Boon Ping Gan, and Rassul Ayani. Preventive what-if analysis in symbiotic simulation. In *Proceedings of the 2008 Winter Simulation Conference*, pages 750–758, 2008.

[8] R. Beckers, J.L. Deneubourga, and S. Gossa. Trails and u-turns in the selection of a path by the ant lasius niger. *Journal of Theoretical Biology*, 159(4):397–415, December 1992.

[9] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE – a FIPA-compliant agent framework. In *Proceedings of the Practical Applications of Intelligent Agents and Multi-agent Systems (PAAM)*, pages 97–108, 1999.

[10] Christian Bierwirth. A generalized permutation approach to job shop scheduling with genetic algorithms. *OR Spektrum*, 17:87–92, 1995.

[11] Piero P. Bonissone, Raj Subbu, Neil Eklund, and Thomas R. Kiehl. Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280, June 2006.

[12] George E. P. Box and Norman R. Draper. *Empirical model-building and response surfaces*. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc., 1986.

[13] Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. Extending the capability concept for flexible BDI agent modularization. In *PROMAS*, pages 139–155, 2005.

[14] S.M. Brennan, A.M. Mielke, D.C. Torney, and A.B. Maccabe. Radiation detection with distributed sensor networks. *Computer*, 37(8):57–59, August 2004.

[15] Paolo Busetta, Nicholas Howden, Ralph Rönnquist, and Andrew Hodgson. Structuring BDI agents in functional clusters. In *ATAL*, pages 277–289, 1999.

[16] V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[17] K. Mani Chandy, Julian J. Bunn, and Annie H. Liu. Models and algorithms for radiation detection. In *Workshop on Grand Challenges in Modeling, Simulation and Analysis in Homeland Security*, 2010.

[18] Chun-Hung Chen and Loo Hay Lee. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific Publishing Co., 2010.

[19] Dan Chen, Boon Ping Gan, Nirupam Julka, Stephen John Turner, Wentong Cai, and Junhu Wei. Evaluating alternative solutions for cloning in distributed simulation. In *Proceedings of the 36th Annual Simulation Symposium*, 2003.

[20] Dan Chen, Stephen John Turner, Wentong Cai, Boon Ping Gan, and Malcolm Yoke Hean Low. Incremental hla-based distributed simulation cloning. In *Proceedings of the Winter Simulation Conference*, pages 386–394, 2004.

[21] Dan Chen, Stephen John Turner, Boon Ping Gan, Wentong Cai, Junhu Wei, and Nirupam Julka. Alternative solutions for distributed simulation cloning. *Simulation*, 79(5–6):299–315, May–June 2003.

[22] P. Christillin. Nuclear compton scattering. *Journal of Physics G: Nuclear Physics*, 12(9):837–851, September 1986.

[23] S.Y.F. Chu, L.P. Ekström, and R.B. Firestone. The lund/lbnl nuclear data search. `http://nucleardata.nuclear.lu.se/nucleardata/toi/`, February 1999.

[24] Robert Cleary and Michael O'Neill. An attribute grammar decoder for the 01 multi constrained knapsack problem. In *Proceedings of the 5th European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 34–45, 2005.

[25] Pierre Collet, Evelyne Lutton, Marc Schoenauer, and Jean Louchet. Take it EASEA. In *Proceeding of the 6th International Conference on Parallel Problem Solving from Nature*, pages 891–901, 2000.

[26] David W. Corne and Joshua D. Knowles. No free lunch and free leftovers theorems for multiobjective optimisation problems. In *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization*, pages 327–341, 2003.

[27] J. Cortial, Charbel Farhat, Leonidas J. Guibas, and M. Rajashekhar. Compressed sensing and time-parallel reduced-order modeling for structural health monitoring using a DDDAS. In *Proceedings of the International Conference on Computational Science*, pages 1171–1179, 2007.

[28] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold Company, New York, January 1991.

[29] Wayne J. Davis. On-line simulation: Need and evolving research requirements. In Jerry Banks, editor, *Handbook of Simulation*, pages 465–516. Wiley-Interscience, 1998.

[30] Anton de Bary. *Die Erscheinung der Symbiose*. Verlag von Karl J. Trübner, Strasbourg, 1879.

[31] Kenneth A. De Jong. *Evolutionary Computation – A Unified Approach*, chapter 6.6, pages 185–188. The MIT Press, 2006.

[32] A.E. Douglas. *Symbiotic Interactions*. Oxford University Press, 1994.

[33] Craig C. Douglas, Jonathan D. Beezley, Janice L. Coen, Deng Li, Wei Li, Alan K. Mandel, Jan Mandel, Guan Qin, and Anthony Vodacek. Demonstrating the validity of a wildfire DDDAS. In *Proceedings of the International Conference on Computational Science*, pages 522–529, 2006.

[34] Glenn R. Drake and Jeffrey S. Smith. Simulation system for real-time planning, scheduling, and control. In *Proceedings of the Winter Simulation Conference*, pages 1083–1090, 1996.

[35] Karl Duncker. On problem solving. *Psychological Monographs*, 58(5 (Whole No. 270)), 1945.

[36] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Sciences*, pages 39–43, 1995.

[37] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[38] H.A. ElMaraghy, I.B. Abdallah, and W.H. ElMaraghy. On-line simulation and control in manufacturing systems. *CIRP Annals-Manufacturing Technology*, 47(1):401–404, 1998.

[39] C.H. Fine. *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*. Perseus Books Group, 1999.

[40] D.B. Fogel. What is evolutionary computation? *IEEE Spectrum*, 37(2):26,28–32, February 2000.

[41] L.J. Fogel, A.J. Owens, and M.J. Walsh. Artificial intelligence through simulated evolution. 1966.

[42] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 3rd edition, 2004.

[43] A. Freville and G. Plateau. Hard 0-1 multiknapsack test problems for size reduction methods. *Investigation Operativa*, 1:251–270, 1990.

[44] Richard Fujimoto, Dell Lunceford, Ernest Page, and Adelinde M. Uhrmacher (editors). Grand challenges for modeling and simulation: Dagstuhl report. Technical Report 350, Schloss Dagstuhl. Seminar No 02351, August 2002.

[45] Richard M. Fujimoto. *Parallel and Distributed Simulation Systems*. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[46] Richard M. Fujimoto, Randall Guensler, Michael Hunter, Hoe Kyoung Kim, Jaesup Lee, John Leonard, Mahesh Palekar, Karsten Schwan, and Balasubramanian Seshasayee. Dynamic data driven application simulation of surface transportation systems. In *Proceedings of the International Conference on Computational Science*, pages 425–432, 2006.

[47] Boon Ping Gan, Lai Peng Chan, and Stephen John Turner. Interoperating simulations of automatic material handling systems and manufacturing processes. In *Proceedings of the Winter Simulation Conference*, pages 1129–1135, 2006.

[48] Joost Garen. Multiobjective job-shop scheduling with genetic algorithms using a new representation and standard uniform crossover. `http://webhost.ua.ac.be/eume/workshops/momh/momh-10.pdf` [last accessed on 2009-01-15].

[49] Mitsuo Gen, Yasuhiro Tsujimura, and Erika Kubota. Solving job-shop scheduling problems by genetic algorithm. In *Proceedings of the IEEE International Confer-*

ence on Systems, Man, and Cybernetics, volume 2, pages 1577–1582, San Antonio, TX, USA, October 1994.

[50] John B. Gilmer Jr. and Frederick J. Sullivan. Alternative implementations of multitrajectory simulation. In *Proceedings of the 1998 Winter Simulation Conference*, pages 865–872, 1998.

[51] John B. Gilmer Jr. and Frederick J. Sullivan. Multitrajectory simulation performance for varying scenario sizes. In *Proceedings of the 1999 Winter Simulation Conference*, pages 1137–1146, 1999.

[52] F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.

[53] Matteo Golfarelli and Stefano Rizzi. Uml-based modeling for what-if analysis. In *DaWaK '08: Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*, pages 1–12, Berlin, Heidelberg, 2008. Springer-Verlag.

[54] Matteo Golfarelli, Stefano Rizzi, and Andrea Proli. Designing what-if analysis: towards a methodology. In *DOLAP '06: Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pages 51–58, New York, NY, USA, 2006. ACM.

[55] Y.C. Ho and D.L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115(3):549–570, December 2002.

[56] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[57] Shell Ying Huang, Wentong Cai, S.J. Turner, Wen Jing Hsu, Suiping Zhou, Malcolm Yoke Hean Low, R. Fujimoto, and R. Ayani. A generic symbiotic simulation

framework. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation (PADS)*, pages 131–131, 2006.

[58] Maria Hybinette. Just-in-time cloning. In *Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS)*, 2004.

[59] Maria Hybinette and Richard Fujimoto. Cloning: A novel method for interactive parallel simulation. In *Proceedings of the Winter Simulation Conference*, pages 444–451, 1997.

[60] Maria Hybinette and Richard Fujimoto. Dynamic virtual logical processes. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS)*, 100–107, 1998.

[61] Maria Hybinette and Richard M. Fujimoto. Cloning parallel simulations. *ACM Transactions on Modeling and Computer Simulation*, 11(4):378–407, October 2001.

[62] International Atomic Energy Agency (IAEA). Inadequate control of world's radioactive sources. Presss Release, June 2002. Available at `http://www.iaea.org/NewsCenter/PressReleases/2002/prn0209.shtml` (last accessed on 2008/12/14).

[63] B. Jarboui, M. Cheikh, P. Siarry, and A. Rebai. Combinatorial particle swarm optimization (cpso) for partitional clustering problem. *Applied Mathematics and Computation*, 192(2):337–345, 2007.

[64] D. F. Jones, S. K. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1–9, February 2002.

[65] Jae-Yoon Jung and James A. Reggia. Evolutionary design of neural network architectures using a descriptive encoding language. *IEEE Transactions on Evolutionary Computation*, 10(6):676–688, December 2006.

[66] Farzad Kamrani. Using on-line simulation in UAV path planning. Licentiate Thesis in Electronics and Computer Systems, KTH, Stockholm, Sweden, 2007.

[67] Farzad Kamrani and Rassul Ayani. Using on-line simulation for adaptive path planning of UAVs. In *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-time Applications*, pages 167–174, Chania, Greece, October 2007.

[68] David Katz and S. Manivannan. Exception management on a shop floor using online simulation. In *Proceedings of the Winter Simulation Conference*, pages 888–896, 1993.

[69] M. Keijzer, J.J. Merelo, G. Romero, and Marc Schoenauer. Evolving objects: A general purpose evolutionary computation library. In *Proceedings of the 5th International Conference on Artificial Evolution*, pages 231–242, 2002.

[70] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[71] Catriona Kennedy and Georgios K. Theodoropoulos. Intelligent management of data driven simulations to support model building in the social sciences. In *Proceedings of the International Conference on Computational Science*, pages 562–569, 2006.

[72] Catriona Kennedy, Georgios K. Theodoropoulos, Volker Sorge, Edward Ferrari, Peter Lee, and Chris Skelcher. AIMSS: An architecture for data driven simulations in the social sciences. In *Proceedings of the International Conference on Computational Science*, pages 1098–1105, 2007.

[73] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[74] Sami Khuri, Thomas Bäck, and Jörg Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing*, pages 188–193, 1994.

[75] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[76] D. Knight, Q. Ma, T. Rossman, and Y. Jaluria. Evaluation of fluid-thermal systems by dynamic data driven application systems - part ii. In *Proceedings of the International Conference on Computational Science*, pages 1189–1196, 2007.

[77] D. Knight, T. Rossman, and Y. Jaluria. Evaluation of fluid-thermal systems by dynamic data driven application systems. In *Proceedings of the International Conference on Computational Science*, pages 473–480, 2006.

[78] J.R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Stanford University, Computer Science Department, 1990.

[79] S. Labov, L. Hiller, K. Nelson, Y. Yao, K. Chandy, A. Liu, M. Wu, and R. Sherbert. Data fusion with distributed mobile detectors in a highly variable background. In *Symposium on Radiation Measurements and Applications*, Ann Arbor, MI, May 2010.

[80] Michael F. L'Annunziata. *Radioactivity – Introduction and History.* Elsevier, 2007.

[81] S. Lawrence. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[82] Peter Lendermann, Malcolm Yoke Hean Low, Boon Ping Gan, Nirupam Julka, Lai Peng Chan, Stephen J. Turner, Wentong Cai, Xiaoguang Wang, Loo Hay Lee, Terence Hung, Simon J.E. Taylor, Leon F. McGinnis, and Stephen Buckley. An integrated and adaptive decision-support framework for high-tech manufacturing and service networks. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the Winter Simulation Conference*, 2005.

[83] Malcolm Yoke Hean Low, Kong Wei Lye, Peter Lendermann, Stephen John Turner, Reman Tat Wee Chim, and Surya Hadisaputra Leo. An agent-based approach for managing symbiotic simulation of semiconductor assembly and test operation. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 85–92, New York, NY, USA, 2005. ACM Press.

[84] Malcolm Yoke Hean Low, Stephen John Turner, Ding Ling, Hai Liang Peng, Lai Peng Chai, Peter Lendermann, and Steve Buckley. Symbiotic simulation for business process re-engineering in high-tech manufacturing and service networks. In *Proceedings of the Winter Simulation Conference*, pages 586–576, 2007.

[85] Marianela Garcia Lozano, Farzad Kamrani, and Farshad Moradi. Symbiotic simulation (S2) based decision support. Methodology report FOI-R-1935-SE, FOI - Swedish Defence Research Agency, February 2006.

[86] Gregory R. Madey, Albert-László Barabási, Nitesh V. Chawla, Marta Gonzalez, David Hachen, Brett Lantz, Alec Pawling, Timothy Schoenharl, Gábor Szabó, Pu Wang, and Ping Yan. Enhanced situational awareness: Application of DDDAS concepts to emergency and disaster management. In *Proceedings of the International Conference on Computational Science*, pages 1090–1097, 2007.

[87] Gregory R. Madey, Gábor Szabó, and Albert-László Barabási. WIPER: The integrated wireless phone based emergency response system. In *Proceedings of the International Conference on Computational Science*, pages 417–424, 2006.

[88] Kumar Mahinthakumar, Gregor von Laszewski, S. Ranji Ranjithan, Downey Brill, Jim Uber, Ken Harrison, Sarat Sreepathi, and Emily M. Zechman. An adaptive cyberinfrastructure for threat management in urban water distribution systems. In *Proceedings of the International Conference on Computational Science*, pages 401–408, 2006.

[89] Amitava Majumdar, Adam Birnbaum, Dong Ju Choi, Abhishek Trivedi, Simon K. Warfield, Kim Baldridge, and Petr Krysl. A dynamic data driven grid system for intra-operative image guided neurosurgery. In *Proceedings of the International Conference on Computational Science*, pages 672–679, 2005.

[90] Jan Mandel, Jonathan D. Beezley, Lynn S. Bennethum, Soham Chakraborty, Janice L. Coen, Craig C. Douglas, Jay Hatcher, Minjeong Kim, and Anthony Vodacek. A dynamic data driven wildland fire model. In *Proceedings of the International Conference on Computational Science*, pages 1042–1049, 2007.

[91] Jan Mandel, Mingshi Chen, Leopoldo P. Franca, Craig J. Johns, A. Puhalskii, Janice L. Coen, Craig C. Douglas, Robert Kremens, Anthony Vodacek, and Wei Zhao. A note on dynamic data driven wildfire modeling. In *Proceedings of the International Conference on Computational Science*, pages 725–731, 2004.

[92] Claudio Mattiussi and Dario Floreano. Analog genetic encoding for the evolution of cirtcuits and networks. *IEEE Transactions on Evolutionary Computation*, 11(5):596–607, October 2007.

[93] James D. McCalley, Vasant Honavar, Sarah M. Ryan, William Q. Meeker, Daji Qiao, Ronald A. Roberts, Yuan Li, Jyotishman Pathak, Mujing Ye, and Yili Hong. Integrated decision algorithms for auto-steered electric transmission system asset management. In *Proceedings of the International Conference on Computational Science*, pages 1066–1073, 2007.

[94] James D. McCalley, Vasant Honavar, Sarah M. Ryan, William Q. Meeker, Ronald A. Roberts, Daji Qiao, and Yuan Li. Auto-steered information-decision processes for electric system asset management. In *Proceedings of the International Conference on Computational Science*, pages 440–447, 2006.

[95] John McCarthy. The inversion of functions defined by turing machines. In C.E. Shannon and John McCarthy, editors, *Automata Studies, Annals of Mathematical Studies*, number 34, pages 177–181. Princeton University Press, 1956.

[96] Bertrand Meyer. *Object-oriented software construction (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[97] John Michopoulos, Panagiota Tsompanopoulou, Elias N. Houstis, and Anupam Joshi. Agent-based simulation of data-driven fire propagation dynamics. In *Proceedings of the International Conference on Computational Science*, pages 732–739, 2004.

[98] John Michopoulos, Panagiota Tsompanopoulou, Elias N. Houstis, John R. Rice, Charbel Farhat, Michel Lesoinne, and Frederic Lechenault. DDEMA: A data driven environment for multiphysics applications. In *Proceedings of the International Conference on Computational Science*, pages 309–318, 2003.

[99] National Science Foundation. DDDAS: Dynamic data driven applications systems. Program Solicitation 05-570, `http://www.nsf.gov/pubs/2005/nsf05570/nsf05570.htm`, 2005.

[100] National Science Foundation. Cyber-physical systems (CPS). Program Solicitation 10-515, http://www.nsf.gov/pubs/2010/nsf10515/nsf10515.htm, 2010.

[101] Allen Newell and Herbert A. Simon. *Human Problem Solving*. Prentice-Hall, Inc., 1972.

[102] Miguel Nicolau and Ian Dempsey. Introducing grammar based extensions for grammatical evolution. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 648–655, 2006.

[103] Object Management Group™. Object constraint language specification. http://www.omg.org/spec/UML/2.2/ [Last accessed on 2010-03-17], May 2006.

[104] Object Management Group™. OMG Systems Modeling Language (OMG SysML™). http://www.sysml.org/specs.htm [Last accessed on 2009-01-30], November 2008.

[105] Object Management Group™. OMG Unified Modeling Language (OMG UML™). http://www.omg.org/spec/UML/2.2/ [Last accessed on 2010-03-10], February 2009.

[106] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, August 2001.

[107] Y.S. Ong, M.H. Lim, and X.S. Chen. Research frontier: Memetic computation – past, present & future. *IEEE Computational Intelligence Magazine*, 5(2), 2010.

[108] Surindar Paracer and Vernon Ahmadjian. *Symbiosis: An Introduction to Biological Associations*. Oxford University Press US, 2000.

[109] Helen Pearson. Genetics: What is a gene? *Nature*, 441(441):398–401, May 2006.

[110] K. Perumalla, R. Fujimoto, T. McLean, and G. Riley. Experiences applying parallel and interoperable network simulation techniques in on-line simulations of military networks. In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, pages 88–95, 2002.

[111] Michele E. Pfund, Scott J. Mason, and John W. Fowler. Handbook of production scheduling. In Jeffrey W. Herrmann, editor, *Semiconductor Manufacturing Scheduling and Dispatching*, volume 89 of *International Series in Operations Research & Management Science*, pages 213–241. Springer New York, 2006.

[112] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm – a novel tool for complex optimisation problems. In *Proceedings of the 2nd I* PROMS Virtual Conference on Intelligent Production Machines and Systems*, pages 454–459, 2006.

[113] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex: A BDI reasoning engine. In R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming*, pages 149–174. Springer Science+Business Media Inc., USA, 2005.

[114] Juergen Potoradi, Ong Siong Boon, Jon W. Fowler, Michele E. Pfund, and Scott J. Mason. Using simulation-based scheduling to maximize demand fulfillment in a semiconductor assembly facility. In *Proceedings of the Winter Simulation Conference*, pages 1857–1861, 2002.

[115] Nicholas J. Radcliffe. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10(4):339–384, December 1997.

[116] R. Rajkumar, Insup Lee, Lui Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *47th ACM/IEEE Design Automation Conference (DAC)*, pages 731 –736, June 2010.

[117] Ingo Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973.

[118] H. Rosoff and D. von Winterfeldt. A risk and economic analysis of dirty bomb attacks on the ports of Los Angeles and Long Beach. *Risk Analysis*, 27(3):533–546, July 2007.

[119] R.Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1997.

[120] Jordi Sabater, Carles Sierra, Simon Parsons, and Nicholas R. Jennings. Using multi-context systems to engineer executable agents. In *ATAL*, pages 260–276, 1999.

[121] Wolfgang Scholl and Joerg Domaschke. Implementation of modeling and simulation in semiconductor wafer fabrication with time constraints between wet etch and furnace operations. *IEEE Transactions on Semiconductor Manufacturing*, 13(3):273–277, August 2000.

[122] Shiyuo Senju and Yoshiaki Toyoda. An approach to linear programming with 0-1 variables. *Management Science*, 15(4):B196–B207, December 1968.

[123] S.P. Sethi, Kwok-Fai Chu, and Houmin Yan. Efficient setup/dispatching policies in a semiconductor manufacturing facility. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 2, pages 1368 –1373 vol.2, 1999.

[124] W. Shih. A branch and bound method for the multiconstraint zero-one knapsack problem. *Journal of the Operational Research Society*, 30(4):369–378, 1979.

[125] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[126] CA Tovey. The honey bee algorithm: A biologically inspired approach to internet server optimization. *Engineering Enterprise, Spring*, pages 13–15, 2004.

[127] Christian Veenhuis, Katrin Franke, and Mario Köppen. A semantic model for evolutionary computation. In *Proceedings of the 6th International Conference on Soft Computing*, 2000.

[128] Christian Veenhuis, Mario Köppen, and Katrin Franke. Hierarchical modelling of evolutionary computation. In *Soft Computing: Methodologies and Applications*, volume 32/2005, pages 97–111. Springer Berlin/Heidelberg, 2005.

[129] H.M. Weingartner and D.N. Ness. Methods for the solution of the multi-dimensional 0/1 knapsack problem. *Operations Research*, 15(1):83–103, 1967.

[130] P.A. Whigham. Grammatically-based genetic programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, 1995.

[131] D.M. Wilkinson. At cross purposes. *Nature*, 412(6846):485, 2001.

[132] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[133] David H. Wolpert and William G. Macready. Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation*, 9(6):721–735, December 2005.

[134] Min-Hao Matt Wu, Annie Hsin-Wen Liu, and K. Mani Chandy. Virtual environments for developing strategies for interdicting terrorists carrying dirty bombs. In *Proceedings of the 5th International ISCRAM Conference*, pages 83–87, Washington, DC, USA, May 2008.

[135] Muh-Cherng Wu, Y. Huang, Y. Chang, and K. Yang. Dispatching in semiconductor fabs with machine-dedication features. *The International Journal of Advanced Manufacturing Technology*, 28:978–984, 2006. 10.1007/s00170-004-2431-x.

[136] Lianqing Xue, Zhenchun Hao, Dan Li, and Xiaoqun Liu. Dongting lake floodwater diversion and storage modeling and control architecture based on the next generation network. In *Proceedings of the International Conference on Computational Science*, pages 841–848, 2007.

[137] M. Yagiura and T. Ibaraki. Use of dynamic programming in genetic algorithms for permutation problems. *European Journal for Operational Research*, 92(2):387–401, 1996.

[138] T. Ye, D. Harrison, B. Mo, B. Sikdar, HT Kaur, S. Kalyanaraman, B. Szymanski, and K. Vastola. Traffic management and network control using collaborative on-line simulation. In *Proceedings of the IEEE International Conference on Communications*, pages 204–209, 2001.

[139] Eugenia Yung. Seven what-if scenarios in post september 11 world. *Algo Research Quarterly*, 4(4):5–10, December 2001.

[140] Fanchao Zeng, Stephen John Turner, Heiko Aydt, and Jusuf Anggono. Symbiotic simulation control in supply chain of lubricant additive industry. In *Proceedings of the 13th International Symposium on Distributed Simulation and Real Time Applications*, 2009.

[141] Huai Zhang, Zhibin Jiang, and Chengtao Guo. Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. *The International Journal of Advanced Manufacturing Technology*, 2008.

[142] K. P. Ziock and W. H. Goldstein. The lost source, varying backgrounds and why bigger may not be better. *Unattended Radiation Sensor Systems for Remote Applications*, 632(1):60–70, October 2002.