PREDICTING THE BEHAVIOR OF ROBOTIC SWARMS IN DISCRETE SIMULATION

by

JOSEPH PAUL LANCASTER JR.

B.S., Kansas State University, 2006

M.S., Kansas State University, 2008

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Abstract

We use probabilistic graphs to predict the location of swarms over 100 steps in simulations in grid worlds. One graph can be used to make predictions for worlds of different dimensions. The worlds are constructed from a single 5x5 square pattern, each square of which may be either unoccupied or occupied by an obstacle or a target. Simulated robots move through the worlds avoiding the obstacles and tagging the targets. The interactions between the robots and the robots and the environment lead to behavior that, even in deterministic simulations, can be difficult to anticipate. The graphs capture the local rate and direction of swarm movement through the pattern. The graphs are used to create a transition matrix, which along with an occupancy matrix, can be used to predict the occupancy in the patterns in the 100 steps using 100 matrix multiplications. In the future, the graphs could be used to predict the movement of physical swarms though patterned environments such as city blocks in applications such as disaster response search and rescue. The predictions could assist in the design and deployment of such swarms and help rule out undesirable behavior.

PREDICTING THE BEHAVIOR OF ROBOTIC SWARMS IN DISCRETE SIMULATION

by

JOSEPH PAUL LANCASTER JR.

B.S., Kansas State University, 2006

M.S., Kansas State University, 2008

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
David Gustafson

Abstract

We use probabilistic graphs to predict the location of swarms over 100 steps in simulations in grid worlds. One graph can be used to make predictions for worlds of different dimensions. The worlds are constructed from a single 5x5 square pattern, each square of which may be either unoccupied or occupied by an obstacle or a target. Simulated robots move through the worlds avoiding the obstacles and tagging the targets. The interactions between the robots and the robots and the environment lead to behavior that, even in deterministic simulations, can be difficult to anticipate. The graphs capture the local rate and direction of swarm movement through the pattern. The graphs are used to create a transition matrix, which along with an occupancy matrix, can be used to predict the occupancy in the patterns in the 100 steps using 100 matrix multiplications. In the future, the graphs could be used to predict the movement of physical swarms though patterned environments such as city blocks in applications such as disaster response search and rescue. The predictions could assist in the design and deployment of such swarms and help rule out undesirable behavior.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Dr. David Gustafson, my advisor and mentor, for years of continual inspiration, guidance, direction, and boundless patience, culminating in this dissertation. I would like to thank Lora Boyer, Jon Tveite, Kathy Zarka, and Kathleen Greene of the McNair Scholars Program at Kansas State University for the invaluable experience that I recieved under their mentorship. I would also like to express my appreciation, on behalf of my family and myself, to my new employer Jay Fredkin and the employees of CABEM Technologies, and to Chris Matthews and Kenneth Sham in particular, for their abundance of understanding and patience in the final days of my candidacy. Finally, I would like to thank the very many professors, teachers, and mentors who excited me about computer science and academia, inspired me over the years, and continue to inspire me today.

Dedication

I dedicate this dissertation to my parents Joseph and Paula Lancaster who taught me the value of hard work and made this work possible by being the cornerstone of my personal and family life throughout my college years. I dedicate this dissertation to my wife Jessie and son Albert who have continually been there for me and provided crucial love, understanding, and support through early mornings, long days, and late nights committed to this dissertation.

Preface

Chapter 4. A version of this material has been published as Lancaster, J. P. and D. A. Gustafson (2013). Predicting the behavior of robotic swarms in search and tag tasks. Procedia Computer Science 20(0), 77-82. Complex Adaptive Systems. http://www.sciencedirect.com/science/article/pii/S1877050913010429

Chapter 1

Introduction

Swarm robotics is a new, highly distributed approach to multi-agent systems (Brambilla et al. 2013). Swarm robotics takes its inspiration from social animals (El Zoghby et al. 2014). Social animals such as colonies of ants, termites, and bees often exhibit what is known as swarm intelligence (El Zoghby et al. 2014). These collectives of very simple animals are able to solve problems that would be impossible for a single individual (El Zoghby et al. 2014). They have evolved such that their interactions with each other and their environment result in complex, seemingly intelligent, behavior. The behavior that emerges from the interactions between individuals and between individuals and their environment is referred to as *swarm intelligence* (Garnier et al. 2007). For example, some termite species build elaborate structures replete with nurseries, horticulture, and ventilation systems (Mueller and Gerardo 2002, Korb 2003). Foraging ants find the shortest path to valuable resources (Bonabeau et al. 1999). The fire ant, Solenopsis invicta, assembles into living rafts to survive floods (Mlot et al. 2011). Slime moulds consisting of amoeba-like cells are able to solve mazes (Nakagaki et al. 2000). Nature has figured out how to do some very impressive things with relatively simple individuals.

Swarm robotics is at the intersection between swarm intelligence and robotics (El Zoghby et al. 2014). A robot swarm is a large group of robots with a collective behavior that results

from their interactions with each other and with their environment (El Zoghby et al. 2014). Interactions between individuals are highly decentralized and the capabilities of the individual are deemphasized. As a result, robot swarms have the advantage of being flexible, scalable, and robust (Brambilla et al. 2013, El Zoghby et al. 2014). Potential applications include warehouse automation (Hsieh and Mather 2012), distributed construction (Hsieh and Mather 2012), firefighting (Parker 1998), mining, minefield clearance (Correll 2007), exploration (El Zoghby et al. 2014), object transport (El Zoghby et al. 2014), object manipulation (El Zoghby et al. 2014), surveillance (Zhu and Yang 2010), locating explosives (Hereford 2010), search and rescue (Parker 1998), security (Parker 1998), cleaning (Correll 2007), painting (Correll 2007), inspection (Correll 2007), maintenance (Parker 1998), plowing (Parker et al. 2003), mowing (Correll 2007), vacuuming, environmental monitoring (Hsieh and Mather 2012), remote sensing (El Zoghby et al. 2014), toxic spill cleanup (Parker 1998), and drug delivery (Hereford 2010).

Unfortunately, developing swarms that do what we want them to do is hard. The behavior that emerges from the interactions between individuals and between individuals and the environment is often non-linear (Bjerknes et al. 2007, El Zoghby et al. 2014, Khaluf et al. 2013a). Swarm design suffers from the inverse problem, meaning that it is hard to know how to produce a controller for a given collective behavior (Berman et al. 2007, Bjerknes et al. 2007). In addition, much work up until recently has involved running lots of resource-intensive simulations or even trial and error experiments with real robots (Muniganti and Pujol 2010). Realistic simulations can be time consuming and experiments with large numbers of physical robots can be impractical (Muniganti and Pujol 2010). Fortunately, recent work on microscopic and macroscopic models has begun to develop methods to understand and quickly predict swarm behavior and even optimize parameters.

Macroscopic models are a family of diverse mathematical representations of collective swarm behavior that have seen some success in the literature (Muniganti and Pujol 2010). A related family of diverse representations, microscopic models, is used to predict swarm

behavior by tracking individual robots, either embodied in high fidelity simulation or by representing them as a collection of interacting stochastic processes. More recently, macroscopic models, have been used to make predictions by tracking average quantities rather than the state of individual robots. The term "macroscopic model" refers to a number of mathematical representations of swarm behavior proposed by a number of different researchers that only track the evolution of the average quantities of a few variables (Brambilla et al. 2013). Fortunately, this can make an otherwise intractible problem tractible (El Zoghby et al. 2014). Macroscopic models represent average swarm behavior (El Zoghby et al. 2014, Massink et al. 2013). They allow accurate predictions (Muniganti and Pujol 2010). They are often parameterized allowing for parameter exploration and optimization (Hamann and Wörn 2007a, Berman et al. 2006). They can assist with design (Berman et al. 2006). They are used in swarm engineering along with model checking in order to reduce the need for testing (Brambilla et al. 2013). They can be accompanied by model checking (Brambilla et al. 2013). This is opposed to the microscopic models that do not scale well due to the fact that they do track each and every member of the collective along with the interactions between those members. For this reason, the microscopic models may not be viable representations for many typical swarms since the number of individuals in these multi-agent systems can easily number in the hundreds, thousands, or more.

In this work, we present a novel macroscopic model for predicting the spatial distribution of robotic swarms using probabilistic graphs (Section 4.4) and occupancy matrices (Section 4.6.1). To our knowledge, the use of probabilistic graphs and occupancy matrices to represent swarm occupancy does not occur in the literature. Furthermore, few existing works explicitly predict the spatial distribution of a robotic swarm. This work is an extension of the SeaTag simulations in (Gustafson and Gustafson 2004 2006). We simulate swarms of robots (Section 4.1.4) that move through grid worlds (Section 4.1) and tag targets (Section 4.1.1) in what we label world executions (Section 4.2.1). Our worlds are partitioned into uniform regions, each having the same square pattern (Section 4.1.2). Our robots have

the Markov property, meaning that each action depends only on the current state and not on past ones. We use graphs and occupancy matrices to predict the spatial distribution of the swarm at each step. The graphs capture how the robots move within and out of the regions. They are simple enough to understand at a glance and when used to create a transition matrix, we can produce a prediction occupancy matrix (Section 4.6) for every step using matrix multiplication. The predictions in our prediction occupancy matrices either roughly contain or accurately track the regions with the highest swarm concentration.

To predict the distribution of robots over the regions in a world execution, we combine many of them into a batch execution (Section 4.2.4). A batch execution comprises one square pattern, a set of parameter values, and a number of world executions determined by the number of trials, which is a parameter of the batch execution. The world executions differ only by the trial number, which is the seed used by the random number generator that robots use to choose actions. In any world execution, a robot's chosen action is based on sensor input, the random number generator, and several probabilities that are parameters of the batch execution. Thus, while the world executions in a batch execution are very similar, it is unlikely that two world executions are the same. By combining a large number of world executions we are able to make predictions about average behavior rather than the behavior in any one world execution. We proceed by creating graphs that capture regional behavior in the batch execution and produce prediction occupancy matrices in turn.

We demonstrate our method on batch executions with square patterns and parameter values chosen such that they present an array of distinct outcomes. For example, in one pattern the swarm moves off to the east and in another the swarm moves off to the west. In one the swarm moves quickly and in another it moves relatively slowly. These differences arise from the choice of square pattern and parameter values. The square pattern and parameter values influence target interference and the degree of non-determinism. They, in turn, effect the rate and direction of swarm movement. Square patterns may create impassible boundaries for robots, giving them no other option than to move east or west. Or,

combined with the parameter values, they may simply encourage them to move east or west. They cause agents to stop, search for targets, or tag targets, all of which take time. Robots can remain trapped by the heat of targets until other robots tag the target(s) responsible or come within sensor range and distract them from the offending target(s). Trapped robots slow the movement of the swarm by not progressing, not tagging, and not being available to distract other robots. This is especially the case since our robots cannot discriminate between robots, targets, and obstacles as they are equipped with only sonar (capable of sensing distance and direction) and an ambient heat sensor. A touch of uncertainty can be helpful in these cases, often encouraging robots to deviate from relatively rigid behaviors and explore. But completely random behavior may cause them to move in circles. In sum, the choice of square pattern and parameter values can give robots a propensity for directed movement or it may produce robots with more or less chaotic movement. We focus on batch executions where the robots move in particular directions at particular rates.

There are potentially important applications for the prediction of swarm movement. In real world robots, such as a large swarm of simple camera-equipped robots sent out to quickly survey an area following a disaster, such a simulation might reveal the direction and rate of exploration. The operators would want to ensure that this precious resource would be used effectively in the early moments following the disaster. Being able to model the swarm and easily expand the area in the prediction would help the operators understand the collective behavior of the swarm and perhaps a bit about how individual robots might deviate from that behavior. Early on, such swarms may be rather unsophisticated, and it would be important to know that these robots would be unlikely to venture into nearby streets or neighborhoods during the search. In another example, supposing we had a swarm of robots sweeping a minefield, we might want the swarm to spread out and sweep in a particular direction, perhaps in the direction of intended troop movement.

In sum, this approach may help to eliminate unwanted behavior in important swarm robotic applications that involve robots that have the Markov property and operate in

uniform environments. The new representation may also replace many lengthy simulations and costly experiments with real robots. Potential applications of the approach include predicting the result using a swarm for tasks such as exploration, firefighting, warehousing, search and rescue, cleaning, inspection, or toxic spill cleanup in environments with regular features such as indoor or urban environments or even rural roads with grid-like features. Our predictions are validated by comparison with simulations.

Recent work is covered in Chapter 2. The hypothesis is covered in Chapter 3. The experimental setup is covered in Chapter 4. The results are covered in Chapter 5. Possible future applications and directions are discussed in Chapter 6. Chapter 7 concludes.

Chapter 2

Literature Review

Swarm robotics is a new approach to the coordination of large numbers of robots (Brambilla et al. 2013). It is a solution to problems that cannot be solved with a single robot (El Zoghby et al. 2014). Robot swarms are flexible, scalable, and robust (Brambilla et al. 2013, El Zoghby et al. 2014). Some of the potential applications include warehouse automation, vacuuming, environmental monitoring, distributed construction (Hsieh and Mather 2012), firefighting, search and rescue (Couceiro et al. 2014, Grayson Grayson), maintenance, security, toxic spill cleanup (Parker 1998), cleaning, painting, mowing, inspection, mining, minefield clearance (Correll 2007), surveillance (Zhu and Yang 2010), locating explosives, drug delivery (Hereford 2010), plowing (Parker et al. 2003), exploration (Couceiro et al. 2014), object transport, remote sensing, and object manipulation (El Zoghby et al. 2014). For a review of swarm robotics, see (Roy et al. 2014, El Zoghby et al. 2014, Abukhalil et al. 2013, Barca and Sekercioglu 2013, Brambilla et al. 2013, Shi et al. 2012, Mohan and Ponnambalam 2009, Bayindir and Şahin 2007, Beni 2005, Şahin 2005). Unfortunately, the development of correct swarms presents many challenges.

2.1 Overview of Approaches to Swarm Robotics

The development of robotic swarms presents many challenges and many approaches have been developed in response. Swarms offer a high level of complexity that make them difficult to design and their behavior difficult to predict. Despite this complexity, robotic swarms offer a simplicity that makes them attractive. Because of this, many approaches are being developed to address this complexity. These approaches include methods of design and implementation; fault tolerance and security; analysis, verification, and testing; and prediction. While many of these contributions are not directly related to the prediction of swarm behavior, which is the primary focus of our work, they represent an overview of the state of the art to which our work is a contribution. Predictive approaches will be discussed in length in Section 2.2 along with sensor-based simulation, microscopic models (Section 2.2.1), and finally macroscopic models (Section 2.2.2). Our contribution is a predictive macroscopic model and we are building up to the macroscopic models.

Developing swarms that do what they are supposed to do is hard (Brambilla et al. 2014). The behavior that emerges from the interactions between individuals and between individuals and the environment is often non-linear (Bjerknes et al. 2007, El Zoghby et al. 2014, Khaluf et al. 2013a). Swarm design suffers from the inverse problem, meaning that it is hard to know how to produce a controller for a given collective behavior (Berman et al. 2007, Bjerknes et al. 2007). In addition, much work up until recently has involved running lots of resource intensive simulations or even trial and error experiments with real robots (Muniganti and Pujol 2010). Realistic simulations can be time consuming and experiments with large numbers of physical robots can be impractical (Muniganti and Pujol 2010). Fortunately, recent work has proposed a number of potential solutions to the problem.

The literature contains a number of approaches to swarm design. The approaches include the design, simulation, implementation, and automatic synthesis of swarm robot control algorithms. Alfio Borzì and Suttida Wongkaew (Borzì and Wongkaew 2015) implement a flocking controller with a leader. Yara Khaluf, Mauro Birattari, and Heiko Hamann (Khaluf

et al. 2014) implement controllers for swarms with soft-deadlines. Luciano Pimenta et al. (Pimenta et al. 2013) implement a swarm controller for navigation in robots using models from fluid dynamics. Claire Gerrard et al. (Gerrard et al. 2013) implement a swarm controller for search in simulation using artificial reaction networks. James Lindsay and Sidney Givigi et al. (Lindsay et al. 2012) and Luis Mendez and Signey Givigi et al. (Mendez et al. 2012) implement flocking controllers in simulation. Sungju Huh et al. (Huh et al. 2013) introduces a programming model to assist in the writing of swarm robot controllers by abstracting communication, synchronization, and parallel processing. Gregory Mermoud, Utkarsh Upadhyay, William Evans, and Alcherio Martinoli (Mermoud et al. 2014) compare analytical top-down numerical optimization and bottom-up evolutionary approaches to controller synthesis in robots. Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari (Francesca et al. 2014) automate the process of controller design by synthesizing probabilistic finite state machines for the e-puck robot. Gianpiero Francesca, Manuele Brambilla, Vito Trianni, Marco Dorigo, and Mauro Birattari (Francesca et al. 2012) and Jorge Gomes and Anders Christensen (Gomes and Christensen 2013) synthesize robot controllers using evolution. Marius Kloetzer and Calin Belta (Kloetzer and Belta 2006) develop a hierarchical control framework that uses a control system and model checking to map high level specifications to provably correct controllers. Alan Winfield et al. (Winfield et al. 2005) formally specify swarm robot systems using linear time temporal logic and prove their correctness. Manuele Brambilla, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo (Brambilla et al. 2012) present a property-driven design approach that uses formal specification and an iterative process to produce models that can be implemented in simulation or robots. Manuele Brambilla, Marco Dorigo, and Mauro Birattari (Brambilla et al. 2014) introduce an approach to the top-down design and also verification of swarms via formal specification and model checking. Heiko Hamann, Thomas Schmickl, Heinz Wörn, and Karl Crailsheim (Hamann et al. 2012) investigate the symmetry breaking behavior of a swarm controller in an aggregation task. S. Kazadi (Kazadi 2009) presents a general

methodology for swarm design.

Some work has contributed to the fault tolerance and security of swarm robotic controllers. Fault tolerance allows a swarm to function in the face of abnormally functioning individuals and fault detection allows swarms to detect and react to those individuals. Danesh Tarapore et al. (Tarapore et al. 2015) implement fault detection in robot swarms using an approach based on a model of the adaptive immune system. Danesh Tarapore et al. (Tarapore et al. 2013) create a controller capable of fault detection that is inspired by the adaptive immune system. Alan Winfield and Julien Nembrini (Winfield and Nembrini 2006) implement fault tolerance using hazards analysis, failure mode and effect analysis, and reliability modeling. Security allows swarms to respond to malicious individuals, those with the purpose of preventing the swarm from achieving its goals. Ian Sargeant and Allan Tomlinson (Sargeant and Tomlinson 2013) introduce a swarm model that accounts for the difference between failed and malicious individuals and that can be used to identify vulnerabilities in different applications. Fiona Higgins et al. (Higgins et al. 2009) provide a survey on security challenges for swarm robotics.

Other work has been clearing the way for the testing, analysis, and verification of swarm systems. Matthew Hosking and Ferat Şahin (Hosking and Sahin 2010) develop a framework for testable swarm systems using agent-in-the-loop simulations. Toshiyuki Yasuda et al. (Yasuda et al. 2013) analyze collective task allocation using behavioral sequence analysis from ethology. Gianpiero Francesca, Manuele Brambilla, Vito Trianni, Marco Dorigo, and Mauro Birattari (Francesca et al. 2012) evolve robot swarms and analyze their decision making behavior using a biological model. Panagiotis Kouvaros and Alessio Lomuscio (Kouvaros and Lomuscio 2015) verify swarm controllers using temporal-epistemic specifications. Their technique is independent of the number of robots but does not allow for collisions. Vain et al. (Vain et al. 2008) present a model checking based proof method for swarms. Panagiotis Kouvaros and Alessio Lomuscio (Kouvaros and Lomuscio 2013) introduce verification via model checking for multi-agent systems, including swarms. Josie Hunter et al. (Hunter et al.

2013) describe a framework for the verification of multi-agent systems that integrates with existing tools such as PRISM, SPIN, and NuSMV. Edmond Gjondrekaj, Michele Loreti, Rosario Pugliese, Fancesco Tiezzi, Carlo Pinciroli, Manuele Brambilla, Mauro Birattari, and Marco Dorigo (Gjondrekaj et al. 2012) introduce the verification of swarms using the formal language KLAIM. Clare Dixon, Alan Winfield, and Michael Fisher (Dixon et al. 2011) present the verification of swarms using temporal logic and model checking. Savas Konur, Clare Dixon, and Michael Fisher (Konur et al. 2010) demonstrate the verification of swarms using probabilistic temporal logic and model checking. Then there is the emergence of swarm engineering, which combines formal specification, design, verification, and testing into one unified discipline (Brambilla et al. 2013, Winfield et al. 2005).

Finally, there are methods for predicting swarm behavior. In one such example, Antons Rebguns et al. (Rebguns et al. 2008) send out scouts to predict success probability. There is also the author's use of metrics to predict swarm performance (Lancaster and Gustafson 2010). However, most prediction methods involve either sensor-based simulation (Section 2.2) or microscopic (Section 2.2.1) or macroscopic models (Section 2.2.2).

2.2 Predicting Swarm Behavior

There are several methods available in the literature for predicting swarm behavior. While there are mathematical approaches similar to the one in (Khaluf et al. 2013), where the central limit theorem is used to predict the number of objects retrieved in a foraging scenario that includes spatial interference, many involve either sensor-based simulation or mathematical modeling. Sensor-based simulation, realistic simulation that includes high-fidelity physics and sensor noise, is often used to observe the behavior of robotic swarms before running real experiments (Brambilla et al. 2013, Shi et al. 2012, Zhu and Yang 2010). Some simulators that show up frequently in the literature include ARGoS (Pinciroli et al. 2012), Webots (Michel 2004), Player/Stage (Gerkey et al. 2003), and Gazebo (Koenig and Howard

2004).

In this work, we are most interested in a diverse family of mathematical representations called macroscopic models (Section 2.2.2) that allow us to predict the collective behavior of swarms of arbitrary size. We begin by introducing the available literature on microscopic models (Section 2.2.1), followed by macroscopic models that do not predict location. Afterward, we discuss existing macroscopic models that predict the location of a swarm and compare and contrast with our approach.

Microscopic and macroscopic models describe the dynamics of swarm robotic systems (Lerman and Galstyan 2004). They can be used to make accurate predictions (Prorok et al. 2011). They are often used to synthesize controllers (Berman 2010). Microscopic models track individuals and their interactions directly, while macroscopic models track high-level dynamics and lack an explicit representation of individuals or their interactions (Brambilla et al. 2013). While microscopic models can provide very accurate predictions, they do not scale well for large numbers of robots due to the explicit tracking of individuals and their interactions (Lerman and Galstyan 2004). Macroscopic models on the other hand provide tractable predictions for large numbers of robots (Lerman et al. 2005, Lerman and Galstyan 2004).

Macroscopic models allow more robots (Kettler and Wörn 2011b, Ingenieurwissenschaften et al. 2008, Hamann and Wörn 2008) and permit longer simulations (Hamann et al. 2008). Further, the fact that they tend to focus on only a few variables contributes to their accuracy and tractability (Lerman et al. 2005, Lerman and Galstyan 2004). They can also enable parameter optimization (Evans et al. 2010, Liu and Winfield 2010). They provide the average or steady-state behavior of a swarm over many simulations (Pace et al. 2013b, Massink et al. 2013, Kettler and Wörn 2011b, Hamann and Wörn 2008, Lerman et al. 2005).

2.2.1 Microscopic Models

Microscopic models capture swarm dynamics by explicitly representing the dynamics of individual robots and their interactions with each other and their environment and allow behavior predictions (Brambilla et al. 2013, Lerman et al. 2005). Many of these are stochastic simulations where each robot is represented by a stochastic event (Lerman et al. 2005). For this reason, they don't scale well to very large systems (Massink et al. 2013). In the following references, unless otherwise noted, the swarms are represented by a series of stochastic events by representing each robot with a probabilistic finite state machine (PFSM). Our macroscopic model approach differs from these microscopic models by using probabilistic graph (Section 4.4) and transition matrix (Section 4.5) representations and tracking averages rather than individual robots and their interactions. In addition, by using macroscopic modeling, our approach scales for arbitrary numbers of robots, unlike these microscopic models.

The following microscopic models track properties of swarms other than location. Our approach differs from these by explicitly tracking the location of robots. In (Massink et al. 2013 2012), the authors predict concensus in a collective decision-making scenario using stochastic modelling in Bio-PEPA. Levent Bayindir and Erol Şahin (Bayindir and Şahin 2009) predict the largest aggregate size and the largest aggregate size ratio in an aggregation task by representing the swarm as a sequence of probabilistic events. Yansheng Zhang et al. (Zhang et al. 2008) predict availability using probabilistic equations. Kjerstin Williams (Williams 2006) predicts swarm performance in a boundary coverage scenario; the collaboration rate in a stick pulling experiment; the average cluster size, the average number of clusters, and the number of active workers in an object aggregation scenario. Nikolaus Correll and Alcherio Martinoli (Correll and Martinoli 2007b 2005) predict encountering rates and the time to completion in a distributed sensing scenario. Alcherio Martinoli and Kjerstin Easton (Martinoli et al. 2004, Martinoli and Easton 2003ab) predict the collaboration rate, influence of a gripping time parameter, team size, stick distribution, and robot density

for a distributed manipulation task. William Agassounon (Agassounon 2003) predicts event residency time, the fraction of informed robots, and the probability of successful message transmission in distributed sensing experiments; the number of robots in different controller states, the number of clusters, the average cluster size, and the number of active workers in non-collaborative distributed manipulation experiments; and the number of robots in different states and the collaboration rate in collaborative distributed manipulation experiments. Alcherio Martinoli and Auke Ijspeert (Martinoli et al. 1999, Martinoli 1999) predict mean cluster size, the size of the largest cluster, the number of clusters, and the time to a single cluster in aggregation experiments. Alcherio Martinoli (Martinoli 1999) predicts the collaboration rate in a collaborative distributed manipulation experiment.

Microscopic models are also used to predict spatial properties of swarms. Michael Rubenstein et al. (Rubenstein et al. 2013) predict the minimum number of agents, performance, scalability, object rotation, and steady state object speed in a collective transport task using a physics based model. Veysel Gazi (Gazi 2013) models the motion dynamics of swarms using Lagrangian dynamics. Amanda Prorok, Nikolaus Correll, and Alcherio Martinoli (Prorok et al. 2011) predict number of times that sites are visited and the time-dependent and time-independent probability to visit a site using a Markov chain on a lattice of sites that represent a tessellation of space in a bounded arena. Spring Berman (Berman 2010) predicts the fraction of a colony at two different sites in a house hunting scenario and the fraction of recruiters over time and part populations and the fraction of assemblies in a swarm robotic assembly system scenario by representing robots as continuous-time Markov chains in chemical reaction networks or using random walk particle tracking. Heiko Hamann, Bernd Meyer, Thomas Schmickl, and Karl Crailsheim (Hamann et al. 2010) predicts the number and density of aggregating robots in a collision-based adaptive aggregation scenario; the target radii, average robot density, and aggregation threshold in a collective perception scenario; the collective velocity in a collective phototaxis scenario; stationary density distributions of robot state, the movement of food in a foraging scenario; and the effect of varying

the number of robots in tree-like aggregation using a stochastic differential equation.

2.2.2 Macroscopic Models

While microscopic models explicitly track each robot and the robot-robot interactions, *macroscopic models* capture swarm dynamics at the collective level (Pace et al. 2013b). As stated above, macroscopic models represent not one, but a range of mathematical representations of collective swarm behavior. Examples from the literature follow. Unless otherwise stated, the authors of the following work used differential equations to abstract the same probabilistic finite state machines that we saw for the microscopic models. These differential equations capture the average behavior over many stochastic simulations of the microscopic models, and lose the ability to capture outlier behavior in the process (Kettler and Wörn 2011b, Hamann and Wörn 2008).

The following works use macroscopic models to track properties other than the location of robots. Our approach differs from these by tracking the location of robots. Manuele Brambilla, Marco Dorigo, and Mauro Birattari (Brambilla et al. 2014) predict the number of robots in eight controller states using a continuous-time Markov chain within a prescriptive modeling and model checking framework. Heiko Hamann, Gabriele Valentini, Yara Khaluf, and Marco Dorigo (Hamann et al. 2014) and Debdipta Goswami and Heiko Hamann (Goswami and Hamann 2014) predict expected swarm behavior in a collective decision making scenario using linear combinations of polynomials. Reina Andreagiovanni, Marco Dorigo, and Vito Trianni (Reina et al. 2014) predict the proportion of agents in three sub-populations in a nest-selection scenario. Jake Taylor-King et al. (Taylor-King et al. 2014) predict the mean time to find the target area, relative mass, and mean exit time using a transport equation and velocity jump processes in a search scenario. Gabriele Valentini, Heiko Hamann, and Marco Dorigo (Valentini et al. 2014) predict decision accuracy and concensus time using stochastic differential equations. Matthias Vigelius et al. (Vigelius et al. 2014) predict decision time and splitting probability using stochastic chemical kinetics

and stochastic differential equations. Jing Zhou et al. (Zhou et al. 2014ba) predict the steady state labor division in a task allocation scenario using two-dimensional Markov processes. They use eigenvalue theory, matrix diagonalization, and a matrix exponential (Zhou et al. 2014a) and closed form expressions based on statistical laws (Zhou et al. 2014b). Heiko Hamann (Hamann 2013 2012) predicts robot efficiency, the correct steady state, and swarm performance in foraging, collective decision making, aggregation, emergent taxis, and density classification scenarios using predictive mathematical models, one of which is an urn model. Martin Wirsing et al. (Wirsing et al. 2013) predict the maximum allowed rest time in a swarm garbage collection task. Mieke Massink, Manuele Brambilla, Diego Latella, Marco Dorigo, and Mauro Birattari (Massink et al. 2013 2012) predict concensus in a collective decision making scenario using ordinary differential equations derived from continuous-time Markov chains that describe the robots and their interactions. Yara Khaluf, Michele Pace, Marco Dorigo, and Fanz Rammig (Khaluf et al. 2013a) predict the expected amount of work and the probability density function of activity times using integrals of birth-death processes. Gabriele Valentini, Marco A Montes de Oca, Mauro Birattari, and Marco Dorigo (Valentini et al. 2013, de Oca et al. 2011) predict concensus using an absorbing Markov chain. Bijan Ranjbar-Sahraei, Gerhard Weiss, and Karl Tuyls (Ranjbar-Sahraei et al. 2013) predict convergence and convergence speed in a stigmergic coverage scenario using Markov chains. Ani Hsieh and William Mather (Hsieh and Mather 2012, Mather and Hsieh 2011) predict mean ensemble behavior in aggregation and task allocation scenarios. William Evans, Grégory Mermoud, and Alcherio Martinoli (Evans et al. 2010) predict the average number of chains in a distributed self-assembly task. Wenguo Liu and Alan Winfield (Liu and Winfield 2010) predict swarm performance in adaptive collective foraging. Heiko Hamann, Bernd Meyer, Thomas Schmickl, and Karl Crailsheim (Hamann et al. 2010) predict the effectiveness of the decision process, likelihood of decisions to be reached and revised, and time to reach concensus in collision-based adaptive aggregation and emergent density classification scenarios using stochastic differential equations and Fokker-Planck theory. Nikolaus Correll

(Correll 2008) predicts the steady-state task distribution in a case study in distributed task allocation. Alan Winfield, Wenguo Liu, Julien Nembrini, and Alcherio Martinoli (Winfield et al. 2008) predict wireless connectivity in an aggregation scenario. Yansheng Zhang et al. (Zhang et al. 2008) predict robot availability using probabilistic equations. Wenguo Liu (Liu 2008) predicts the number of uncollected food items, the net energy of the swarm, the time for the swarm to complete the task, and the optimal population of the swarm to finish the task in a foraging scenario. Heiko Hamann and Heinz Wörn (Hamann and Wörn 2007a) predict the flow of food in a foraging scenario using partial differential equations. Onur Soysal and Erol Şahin (Soysal and Şahin 2007) predict the final aggregation distributions in an aggregation scenario using the partition concept from number theory. Wenguo Liu, Alan Winfield, and Jin Sa (Liu et al. 2007) predict the instantaneous net energy of a swarm and the number of robots in each of the controller states in a foraging scenario. Nikolaus Correll and Alcherio Martinoli (Correll and Martinoli 2007b) predict the likelihood of convergence and the average number of robots per cluster in an aggregation scenario using a Markov dynamical system for every individual and differential equations to capture average behavior. Nikolaus Correll (Correll 2007) predicts the time to complete coverage in distributed boundary coverage scenarios. Jan Dyre Bjerknes, Alan Winfield, and Chris Melhuish (Bjerknes et al. 2007) predict the distance covered and swarm velocity in an emergent taxis scenario by constructing simple equations based on microscopic considerations. Kristina Lerman, Chris Jones, Aram Galstyan, and Maja Matarić (Lerman et al. 2006) predict the number of robots allocated to two tasks in a multi-foraging scenario. Kjerstin Irja Williams (Williams 2006) predicts swarm performance in a boundary coverage scenario; the collaboration rate in a stick pulling scenario; and the average cluster size, average number of clusters, and number of active workers in an object aggregation scenario. Nikolaus Correll and Alcherio Martinoli (Correll and Martinoli 2005 2007a) predict encountering rates and the time to completion in a distributed sensing scenario. They use regular structures similar to our square patterns. Kristina Lerman and Aram Galstyan (Galstyan and Lerman 2005, Lerman and Galstyan

2003) predict the number of robots allocated to two tasks in an adaptive foraging scenario. Kristina Lerman, Alcherio Martinoli, and Aram Galstyan (Lerman et al. 2004) predict the collaboration rate in a distributed manipulation scenario and the mean cluster size and time to completion in a simplified foraging scenario. Kristina Lerman and Aram Galstyan (Lerman and Galstyan 2004) predict the collaboration rate in a collaboration manipulation scenario. Christopher Vernon Jones, Maja Matarić, Kristina Lerman, et al. (Jones et al. 2004) predict the probability that a swarm will execute a task using a Bayesian macroscopic modeling approach. Alcherio Martinoli, Kjerstin Easton, and William Agassounon (Martinoli et al. 2004, Martinoli and Easton 2003ab) predict the collaboration rate and influence of a gripping time parameter, team size, stick distribution, and robot density for a distributed manipulation task. William Agassounon, Alcherio Martinoli, and Kjerstin Easton (Agassounon et al. 2004) predict average cluster size in an aggregation scenario. Kristina Lerman (Lerman 2004) predicts the collaboration rate in an adaptive collaboration scenario. Chris Parker et al. (Parker et al. 2003) predict nest growth and average nest radius in a collective construction scenario using Markov chains and a predictive algorithm constructed under physical and geometrical considerations. William Agassounon (Agassounon 2003) predicts residency time, the fraction of informed robots, and the probability of successful message transmission in distributed sensing experiments; the number of robots in different controller states, the number of clusters, average cluster size, and the number of active workers in non-collaborative distributed manipulation experiments; and the number of robots in different states and the collaboration rate in collaborative distributed manipulation experiments. Kristina Lerman and Aram Galstyan (Lerman and Galstyan 2002) predict collection time and robot efficiency in foraging scenarios. William Agassounon and Alcherio Martinoli (Agassounon and Martinoli 2002) predict average cluster size, the fraction of active robots, and task stimulus in an aggregation experiment. Kristina Lerman, Aram Galstyan, Alcherio Martinoli, and Auke Ijspeert (Lerman et al. 2001) predict the collaboration rate in a collaborative manipulation experiment. Kristina Lerman and Aram Galstyan (Lerman and

Galstyan 2001) predict the distribution of coalitions in a coalition formation scenario, the collaboration rate in a collaborative distributed manipulation scenario, and the efficiency per robot in a foraging scenario. Aude Billard, Auke Ijspeert, and Alcherio Martinoli (Billard et al. 1999) predict the minimal time delay for learning all locations in an adaptive exploration scenario using a set of probabilistic equations. Scott Jantz et al. (Jantz et al. 1997) predict collision frequency in an effusion scenario using the kinetic theory of gasses.

Finally, macroscopic models have been used to predict swarm location in recent work. Similar to our approach, these works use macroscopic models. And similar to our approach, these works predict the location of a swarm over time. A single simulation of these macroscopic models is enough to make a prediction. In our approach, a single sequence of matrix multiplications is enough to predict the distribution of the swarm in all of the steps over many different simulations. The robots in these works are simple, reactive, and Markovian (Prorok et al. 2011, Hamann and Wörn 2008 2007b). They all focus on only a few parameters and are intended to provide fast and accurate qualitative and quantitative results (Hamann et al. 2008).

Unlike any of the other approaches, our approach uses probabilistic graphs (Section 4.4) and transition matrices (Section 4.5) to make predictions. Our approach makes use of discrete grid worlds (Section 4.1) where most of the other approaches use continuous space, although most of those discretize the space in some way (Prorok et al. 2011, Ingenieurwissenschaften et al. 2008, Hamann et al. 2008). In the other approaches, appropiate parameter values must be found for their models (Ingenieurwissenschaften et al. 2008, Kettler and Wörn 2011a). However, when we evolve graphs, in some cases, we must run many world executions (Section 4.2.1) in a batch execution (Section 4.2.4) to obtain a data set (Section 4.2.4) before we can train our graphs.

Fidel Aznar et al. (Aznar et al. 2014) predict the probable location of a swarm at time t using the Fokker-Planck equation and an iterative process. They discretize the world to 100x100 units. The Fokker-Planck equation requires a single point of origin. Robots

in our method, in constrast, can start anywhere in the world as we need only specify the initial distribution using the regional proportions in the initial occupancy matrix. Michele Pace, Mauro Birattari, and Marco Dorigo (Pace et al. 2013ab) predict swarm position using random finite set theory and measure valued recursions. More specifically, they study the dynamics of spatial diffusion of micro-robots injected into an absorbing medium and motion toward a light source in a complex environment. They also obtain macroscopic descriptions of flocking and aggregating swarms that include feedback. Similar to our approach, they are able to explicitly predict the spatial distribution of the swarm by step. However, they assume that the robots are uniformly distributed. They also do not allow for spatial interference. We allow spatial interference between robots and between robots and the targets and obstacles in environment. There is also feedback between the robots and the environment in our models unlike in some of their models (Pace et al. 2013a). Amanda Prorok, Nikolaus Correll, and Alcherio Martinoli (Prorok et al. 2011) predict spatial distribution over time using a Fokker-Planck diffusion model and describe the dynamics using the Fokker-Planck equation. Like us, they use regular structures similar to our square patterns. They also discretize their worlds into a grid structure and describe how robots move between the different areas. Their random walking agents are similar to the usually random movements of our robots. And their models suffer from initial conditions and noise along the edges of the world similar to ours. However, they assume that their worlds are free from obstacles. In constrast, our worlds are full of obstacles, although we assume that they are placed in a regular pattern. Alexander Kettler and Heinz Wörn (Kettler and Wörn 2011ab) predict the average spatial position and velocity of a robot swarm in two dimensions in dispersion (with obstacle avoidance and with and without gradient descent) and collective clustering scenarios using an integer-differential equation based on the Boltzmann equation from statistical physics. They explicitly track the spatial distribution like we do. On the other hand, they are tracking more than just the spatial distribution, such as velocity, while we are only tracking the spatial distribution. And unlike our robots, their robots employ messaging. They also validate

their models with real robots and we leave this to future work. Spring Berman (Berman 2010) predicts the population fractions at two sites using chemical reaction networks and the chemical master equation. Their discrete worlds are similar to our own. And similar to our models, their models suffer from initial conditions and noise along the edges of the world. They also discretize their worlds into a grid structure and describe how robots move between the different areas. However, they use two areas while we use 50 and 100. Heiko Hamann (Ingenieurwissenschaften et al. 2008) predicts the number and density of aggregating robots in a collision-based adaptive aggregation scenario; the target radii, average robot density, and aggregation threshold in a collective perception scenario; the collective velocity in a collective phototaxis scenario; stationary density distributions of robot state and the flow of food in a foraging scenario with pheromones; and the effect of varying the number of robots in tree-like aggregation using Lengevin and Fokker-Planck equations. They use regular structures similar to our square patterns. Their models suffer from intractabilty issues. In our case, the sizes of our transition matrices are bound by memory. Unlike this work, they also work with inhomogenous space, their models account for drift, and their models are capable of tracking more than just the distribution of the swarm in the world. They also assume that their robots are uniformly distributed. Thomas Schmickl, Heiko Hamann, Heinz Wörn, and Karl Crailsheim (Schmickl et al. 2009) predict the density of robots in designated aggregation areas using Lengevin and Fokker-Planck equations. They are using two aggregation areas whereas we predict the density of robots in 50 and 100 regions. Heiko Hamann, Heinz Wörn, Karl Crailsheim, and Thomas Schmickl (Hamann et al. 2008) predict the spatial distribution of a robotic swarm aggregating at a light source using compartment models and partial differential equations. Their models suffer from initial conditions and noise along the edges of the world similar to ours and they also discretize their worlds into a grid structure and describe how robots move between the different areas. Their models suffer from problems with intractabilty. In our case, the sizes of our transition matrices are bound by memory. Unlike us, they account for drift and validate their models with

real robots. They also assume that the number of robots is fixed or infinite over the course of the simulation. We use a finite number of robots and their numbers may increase and decrease as they enter and leave over the course of a world execution. Heiko Hamann and Heinz Wörn (Hamann and Wörn 2008 2007b) predict collective velocity in an emergent phototaxis scenario with collision avoidance and coherence preservation. They also predict target radii, aggregation threshold, number of robots aggregated at target areas, and, like us, the robot density in a collective perception scenario. They use Lengevin and Fokker-Planck equations. Their random walking agents are similar to the usually random movements of our robots. However, their agents employ messaging while ours do not. Spring Berman, Ádám Halász, Vijay Kumar, and Stephen Pratt (Berman et al. 2006) predict the fraction of a colony at two sites. We share the results of predicting the fraction of our swarm in 50 to 100 regions. Kristina Lerman, Aram Galstyan, Maja Matarić, and Tad Hogg (Lerman et al. 2005, Galstyan et al. 2005) predict the evolution of robot density in one dimension. We predict the evolution of robot density in two dimensions.

Next, we provide our hypothesis in Chapter 3 before detailing our methodology in Chapter 4 and presenting our results in Chapter 5.

Chapter 3

Hypothesis

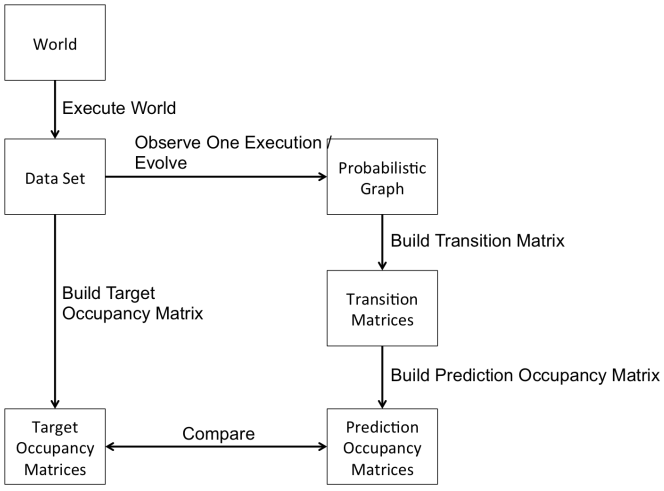

Figure 3.1: *Complete process and data structures.*

We can predict the distribution of robots over the regions of a world over a batch execution using graph representations of local behavior. An overview of the prediction and validation process is shown in Figure 3.1. The figure shows the major data structures and how they are produced from each other. Ultimately, predictions are made about the distribution of robots (Section 4.1.4) in the regions of a world (Section 4.1) in a batch execution (Section 4.2.4).

The predictions take the form of prediction occupancy matrices (Section 4.6) that specify

a proportion of the robots for every region in each step. The prediction occupancy matrices are produced using one to three probabilistic graphs (Section 4.4), each capturing the movement of robots within and out of some of the regions by way of the edge probabilities. The graphs may be generated manually by observing a world execution (Section 4.2.1) or by training on a data set using a genetic algorithm. The prediction occupancy matrices are obtained from the graphs by the generation of a transition occupancy matrix (Section 4.5) from the graphs and successive multiplications of the transition matrix with a linear occupancy matrix (Section 4.6.1) representing the vertex-wise occupancy of a step. For each step, a conversion of the linear occupancy matrix into a two dimensional prediction occupancy matrix yields a prediction for the step.

The predictions are validated using target occupancy matrices (Section 4.3) that provide the regional occupancy for each step in the batch execution. Target occupancy matrices are produced through the batch execution. A batch execution involves one or more executions of an initial world in which the moves of the robots are recorded into a data set (Section 4.2.4). Since a data set may contain the data for more than one world execution, it represents many world executions, each one with its own unique random number generator seed. A data set will contain many world executions unless the probabilities in the paramater values (Section 4.2.4) are such that the batch execution is deterministic. It is this data set that is used to generate the target occupancy matrices for each step. To validate a prediction occupancy matrix, it is compared with the target occupancy matrix for the same step using the predictive value metrics (Section 4.7), an objective measure of similarity between the two matrices.

Chapter 4

Methodology

In this chapter, we describe our methodology in detail. We elaborate on how to construct and execute worlds, produce data sets, and represent the data with target occupancy matrices in Sections 4.1 - 4.3. The first three data structures in our process are shown on the left side of Figure 3.1, which provides an overview of our methodology. The target occupancy matrices serve as our ground truth and our goal is to predict them.

The other three data structures are the probabilistic graphs, transition matrices, and prediction occupancy matrices that we use to make our predictions. These are discussed in Sections 4.4 - 4.6. These data structures are shown on the right side of Figure 3.1. The prediction occupancy matrices are our prediction data structure. We want them to be as close to the target occupancy matrices as possible.

Finally, we discuss our predictive value metric, which is an objective measure of the closeness of our prediction occupancy matrices to our target occupancy matrices. The metric is seen in Figure 3.1 as the bi-directional arrow. The predictive value metric is discussed in Section 4.7.

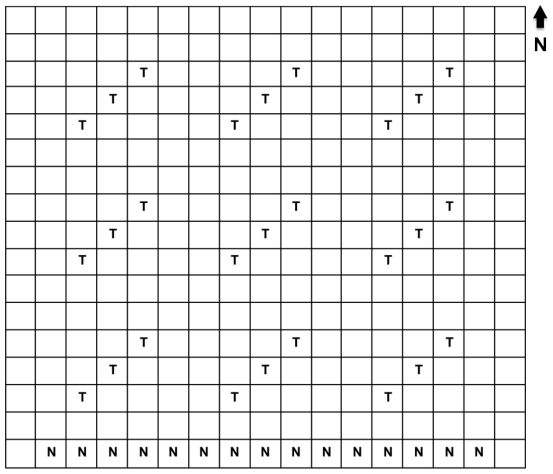

Figure 4.1: *Robots face north in the outer ring of squares and targets are untagged in a 17x17 world.*

4.1 Worlds

We place robots, targets, and obstacles in borderless, two-dimensional grid *worlds*. Robots, targets, and obstacles are collectively known as objects (Section 4.1.1). An example initial world is shown in Figure 4.1. The edges of the world are not detectable by sensors, are not obstacles to movement, and robots that move beyond the edges never return. The grid squares adjacent to the edges do not contain targets or obstacles. Each world is constructed (Section 4.1.3) by partitioning all but the outermost squares into uniform regions and placing objects into each region according to a single 5x5 square pattern (Sections 4.1.2). When referring to worlds, *north* is at the top, *south* is at the bottom, *east* is at the right side, and *west* is at the left side. Worlds come in two sizes: *small*, with a height of 52 squares and a width of 27 squares; and *large*, with a height of 52 squares and width of 52 squares. Note that for a small world, $world\_width = 27$ and $world\_height = 52$. The world sizes are shown in Table 4.1. Worlds are the first data structure in our process shown in Figure 3.1 and ultimately we want to make predictions about the behavior of the robots in them. We will begin to create our other data structures from them in Section 4.2.

Table 4.1: *The two world sizes and their respective dimensions.*

	Width		Height	
	Squares	**Regions**	**Squares**	**Regions**
Small	27	5	52	10
Large	52	10	52	10

T	A target.
O	An object.
N	Robot oriented north.
S	Robot oriented south.
E	Robot oriented east.
W	Robot oriented west.
	Unoccupied.

Table 4.2: *List of world objects.*

4.1.1 World Objects

Targets, obstacles, and robots are all collectively known as *objects*. A world square can be occupied by at most one object in any given step. The majority of squares are *unoccupied*. *Targets*, represented in a square by the character 'T', are immobile, emit heat, and when tagged (when robots collide with them) they transition into obstacles. *Obstacles*, repesented by the character 'O', are immobile and do not emit heat. *Robots* begin in the bottom row of squares, outside of the regions, and face north. Robots are mobile and are represented by one of four characters 'N', 'S', 'E', or 'W' in a given step, referring to their *orientation* of *north*, *south*, *east*, or *west*, respectively. As shown in Figure 4.1, robots are not placed in the two bottom corner squares and so the total number of robots in an initial world is given by $num\_robots = world\_width - 2$. World objects are summarized in Table 4.2.

4.1.2 Square Patterns

Worlds are created from one of several 5x5 *square patterns*. For simplicity, each world uses only one pattern. Each square in a pattern is either unoccupied, occupied by an obstacle

(O), or a target (T). The square patterns that we use are shown in Figure 4.2. Their names reflect their effects. The worlds that we built from these patterns are shown in Appendix A. The use of only one pattern is intended to simplify prediction by reducing the complexity of the problem in the early stages of the investigation.

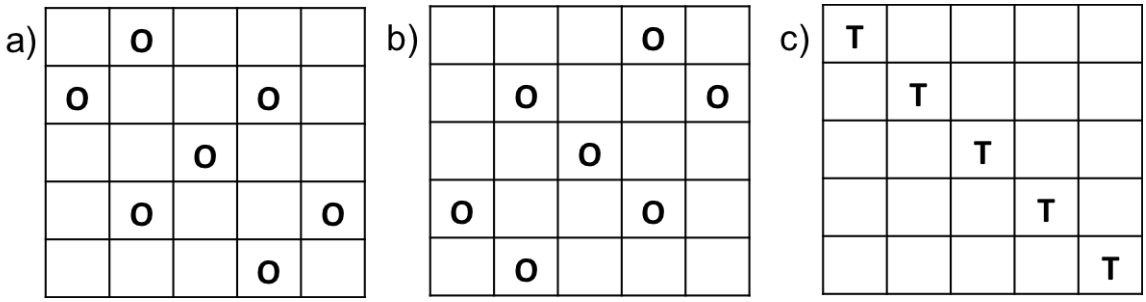

Figure 4.2: *Patterns (a) East, (b) West1, (c) West2.*

4.1.3 Constructing the Worlds

We build each small world from 50 5x5 square regions and each large world from 100 5x5 square regions. Each *region* is a 5x5 pattern with the origin in the bottom left square so that the bottom left and top right squares are at the ordered pairs (0,0) and (4,4), respectively, as exemplified in Figure 4.4. We group the regions into 5 *columns* and 10 *rows* in the small worlds and 10 columns and 10 rows in the large worlds. The regions are labeled by increasing roman numerals along the axes with the origin in the south-west corner. For example, the bottom left and top right regions are at coordinates (I, I) and (V, X), and (I, I) and (X, X) in the small and large worlds, respectively. Figure 4.3 shows a partitioned world with labeled rows and columns.

4.1.4 Robots and Sensors

A robot's (Section 4.1.1) action at any step is conditioned, in part, upon sensor input. Robots are able to sense the direction of objects, including other robots, with *sonar*. They

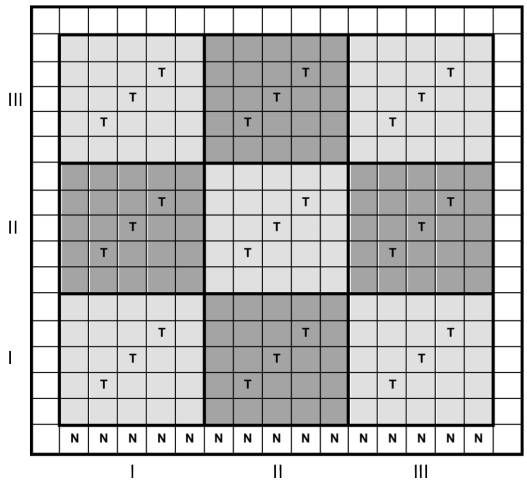

Figure 4.3: *An example 17x17 world divided into 9 5x5 regions.*

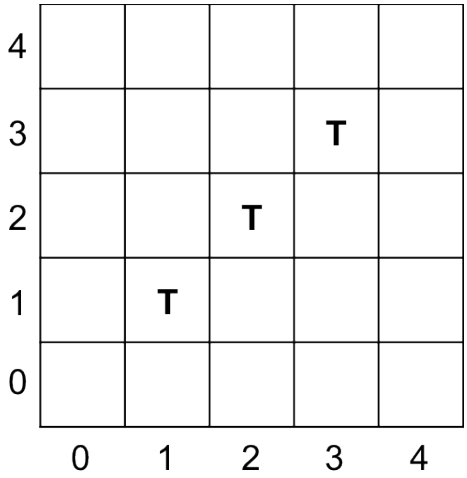

Figure 4.4: *A region with targets located at ordered pairs (1,1), (2,2), and (3,3), respectively.*

sense the presence of targets by sensing their ambient heat with a *heat sensor*. The robots are unable to determine the direction of the heat. The `sonar_range` and `heat_range` are parameters of a batch execution (Section 4.2.4). The detectable sensor region for a robot with a sensor range of 3 is shown in Figure 4.5. In addition to the ranged sensors, robots are able to sense the direction of light shown from the north end of the world from any range using a *photosensor* in a move called *phototaxis*. An action is chosen by the agent function

(Section 4.1.5).

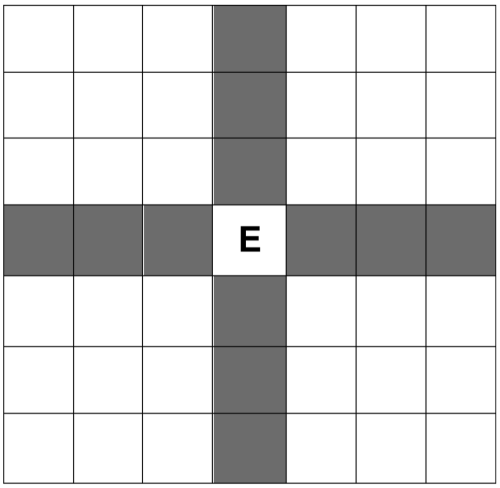

Figure 4.5: *The square region of detection for a robot with a sensor range of 3.*

4.1.5 The Agent Function

The agent function determines the action that a robot will take using the available sensor readings, nested rules, and a function $roll(p) = rng.rand() < p$, where $rng = Random(t)$ is a random number generator seeded by the trial number $t < num\_trials$. The `trial_number` (Section 4.2.4) is unique to each world execution. The rules are tested in order. If a rule fires and it contains nested rules, those nested rules are tested in order. If a rule fires and it contains an action then the action is taken and, unless the action is a `goto`, the turn is ended. Only one branch and associated action is taken per turn but if a nested rule must backtrack the rule search may move from an action to another rule using a `goto` operation.

For each *rotation* or *scan* that occurs in the rules, the robot must choose a direction. The direction is determined using the `right_turn_probability` (`rtp`), which is a parameter of the batch execution. To use a probability p such as the `right_turn_probability`, we use our function $roll(p)$. In the case of `right_turn_probability`, the rotation is clockwise if and only if $roll(right\_turn\_probability)$. Most rotations are 90 degrees. The only exception

is a rotation to face the first detected object in the presence of heat in which case the rotation could be 180° or 270°.

In addition to the `right_turn_probability`, the follow, phototaxis, and lateral phototaxis actions have their own probabilities. While the `right_turn_probability` is intended to add uncertainty through rotations, the follow, phototaxis, and lateral phototaxis moves are intended to decrease uncertainty in robot behavior. They decrease uncertainty by giving robots a chance to move in a directed manner, either by moving with the obstacles in the environment or toward the north end of the world. This is in constrast to normal rotations, such as those taken in most collisions, where the robot simply turns right or left with `right_turn_probability`. Such simple moves tend to lead robots in circles rather than in any particular direction.

The follow, phototaxis, and lateral phototaxis probabilities are used to decide whether a follow, phototaxis, or lateral phototaxis move is taken, respectively. The follow probability (`fp`) is so named because robots engaging in the behavior follow corners. A robot may turn with corners if $roll(follow\_probability)$. The phototaxis probability (`pp`) is named as such because it mimics the similarly named light following behavior in living organisms. If facing south, a robot may turn 90° in the direction of the light if $roll(phototaxis\_probability)$. The east-west lateral phototaxis probability (`lpp`) was given its name only to distinguish it from the probability of the south facing phototaxis moves. If facing east or west, a robot may turn to face north if $roll(lateral\_phototaxis\_probability)$. Different combinations of these three moves, the `heat_range` and `sonar_range` (Section 4.1.4), and the square patterns (Section 4.1.2) lead to very different behavior over time. We took advantage of said combinations to design batch executions where swarms moved in particular directions and rates (Section 4.2).

The rules are as follows:

1. Robot detects heat ⇒

    (a) Scan (use *right_turn_probability*) detects an object ⇒

i. First detected object not ahead

Turn to face the first object in the order scanned.

ii. True (an object detected ahead) ⇒ Goto 2.

(b) True (no object detected) ⇒ Goto 2.

2. True (no heat or no obstacle or obstacle ahead) ⇒

(a) Robot is in collision ⇒

i. Robot is unobstructed on one side, $roll(follow\_probability)$ ⇒

- Tag if target, then turn toward the open side.

ii. True ⇒ Tag if target, Rotate (use $right\_turn\_probability$).

(b) True (Not in collision) ⇒

i. Robot faces south, unobstructed sides, $roll(phototaxis\_probability)$ ⇒

- Rotate (use $right\_turn\_probability$).

ii. Faces south, obstructed on one side, $roll(phototaxis\_probability)$ ⇒

- Turn to face the open side.

iii. Faces east/west, unobstructed north, $roll(lateral\_phototaxis\_probability)$ ⇒

- Turn to face north.

iv. True ⇒ Move forward a single square.

The complete pseudocode for the agent function is provided in Figures 4.6 and 4.7. Execution begins with the `agent_turn` function.

4.2 Data Collection

We execute worlds (Section 4.2.1) using software we have developed (Section 4.2.2) to collect robot paths (Section 4.2.3) into data sets (Section 4.2.4). This step in the process is

```
def move_roll()
    if (obstructed_left() xor obstructed_right()) and roll(follow_probability)
        obstructed_left()
    else
        roll(right_turn_probability)
    end
end

def phototaxis_move()
    move = false
    if not obstructed_left() and not obstructed_right()
       if roll(phototaxis_probability)
          if roll(right_turn_probability)
             rotate_clockwise()
          else
             rotate_counterclockwise()
          end
          move = true
       end
    elsif not obstructed_left()
       if roll(phototaxis_probability)
          rotate_clockwise()
          move = true
       end
    elsif not obstructed_right()
       if roll(phototaxis_probability)
          rotate_counterclockwise()
          move = true
       end
    end
    move
end

def lateral_phototaxis_move()
    result = not occupied_light_direction()
    if result and roll(lateral_phototaxis_probability)
       if lightsource_detected_left()
          rotate_clockwise()
       elsif lightsource_detected_right()
          rotate_counterclockwise()
       end
    end
end
```

Figure 4.6: *Pseudocode for the agent function.*

```
def scan()
   did_move = false
   clockwise = roll(right_turn_probability)
   if clockwise
      scan_result = scan_clockwise(heat_range, sonar_range)
   else
      scan_result = scan_counterclockwise(heat_range, sonar_range)
   end
   if scan_result.heat_detected and scan_result.object_detected
      did_move = rotate_to_face_dir(scan_result.detected_object_direction)
   end
   did_move
end

def move()
    if obstructed_ahead()
       attempt_tag()
       if at_border()
          leave()
       elsif move_roll()
          rotate_clockwise()
       else
          rotate_counterclockwise()
       end
    else
        did_move = false
        if lightsource_detected_behind()
           did_move = phototaxis_move()
        elsif lightsource_detected_right() or lightsource_detected_left()
           did_move = lateral_phototaxis_move()
        end
        if not did_move
           move_forward()
        end
    end
end

def agent_turn()
    did_move = scan()
    if not did_move
       move()
    end
end
```

Figure 4.7: *Pseudocode for the agent function (continued).*

Table 4.3: *Complete list of parameters and the corresponding values used.*

`num_trials`	The number of world executions, each with a different seed.	40000
`max_steps`	The maximum number of steps in the world execution. The world execution ends when all targets are tagged or when the maximum number of steps is reached.	100
`heat_range`	The heat detection range.	$[1, 3]$
`sonar_range`	The sonar range.	$[3, 5]$
`phototaxis_probability`	The probability of turning toward the light when facing south.	1.0
`lateral_phototaxis_probability`	The probability of turning toward the light when facing east or west.	1.0
`follow_probability`	The probability of turning toward the lone open side upon collision.	$[0.0, 1.0]$
`right_turn_probability`	The probability of a right turn or clockwise scan.	$[0.0, 1.0]$

represented by the arrow labeled "Execute World" in Figure 3.1. The paths are a representation of all of the moves a robot makes in the regions of a world execution and slices the executions by robot rather than step. The data sets are used to generate target occupancy matrices (Section 4.3) which serve as our ground truth. They are also used to generate path length histograms (Section 4.3.3) and are a source of statistics from which some of our graphs are produced (Section 4.4.3).

4.2.1 World Execution

World execution consists of taking an initial world (Section 4.1) from step to step for up to `max_steps` using the agent function (Section 4.1.5). A robot can run for fewer than `max_steps` steps if all targets are tagged or all robots leave the world. Each world execution has a unique `trial_number` (Section 4.2.4) that is used to seed the random number generator that the agent function uses to determine the direction of each robot scan and rotation. The robots take turns acting and are not guaranteed to act in any particular order. A step ends after every robot has acted once.

4.2.2 Software

We create, edit, and visualize square patterns (Section 4.1.2) and initial worlds and execute the worlds using software that we developed. We choose the parameter values and either

view the executions in a graphical user interface or run them in batches (Section 4.2.4). When viewed in a GUI window, we have the choice of stepping forward and backward manually or supplying a time interval and stepping forward automatically. Figure 4.8 shows a world execution being setup in our software environment. Select screenshots of actual graphical world executions are shown in Appendix B.

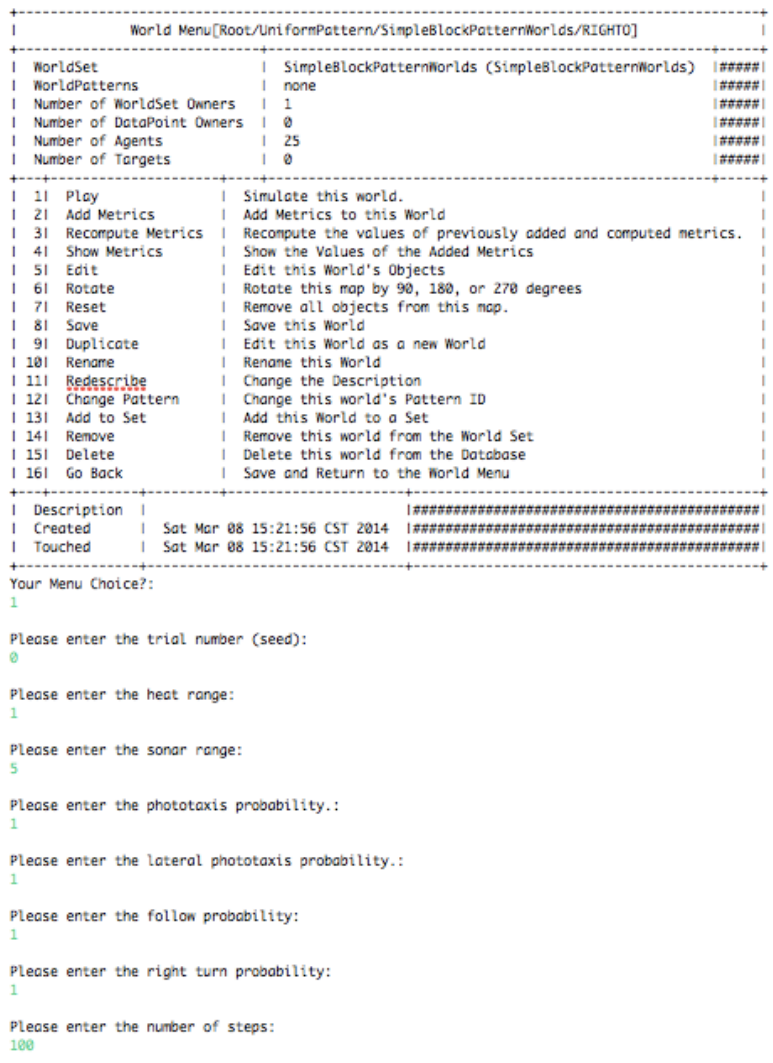

Figure 4.8: *Setting up a world execution using our software.*

4.2.3 Robot Paths

A *path* captures the sequence of occupied regions, occupied squares within the regions, and the orientations of a robot from the step in which it enters a region from a border square. Formally, a path $PATH$ is an initial step s_{PATH} and a location sequence $\langle (p_i, q_i, r_i) \rangle$ where p_i is the position of the occupied region, q_i is the position of the occupied square within the region at position p_i, and o_i is the orientation in the square at q_i at step $s_{PATH} + i$. We refer to p_i as the *global position* and q_i as the *local position*. A location sequence begins with the location (p_0, q_0, r_0) at step s_{PATH} when the robot first leaves the edge squares and enters the regions. A location sequence ends either when the world execution ends or when the robot leaves a region for the edge squares (from which it can never return), whichever comes first. We use the paths as a foundation for rigorous treatment of target occupancy matrices (Section 4.3) and path length histograms (Section 4.3.3). Figure 4.9 shows the locations of two robots in an example world. One robot is located at location $((II, III), (2, 0), W)$ with global position (II, III), local position $(2, 0)$, and orientation West. As an example, consider the following path from the data set (Section 4.2.4) for batch execution "East" (Section 5.1) where a robot entered the world at local position $(2, 0)$ in global position (IV, I) in step 1, moved north in one step, turned right in the next, and moved east for three steps before exiting the world in step 6:

$$\langle 1, \langle ((IV, I), (2, 0), N), ((IV, I), (2, 1), N), ((IV, I), (2, 1), E), ((IV, I), (3, 1), E), ((IV, I), (4, 1), E) \rangle \rangle$$

The corresponding path from the actual data set is parsed by our software:

$$\texttt{1|4\%0 : 2, 0; U\%2, 1; U, R} * \texttt{\%3, 1; R\%4, 1; R}$$

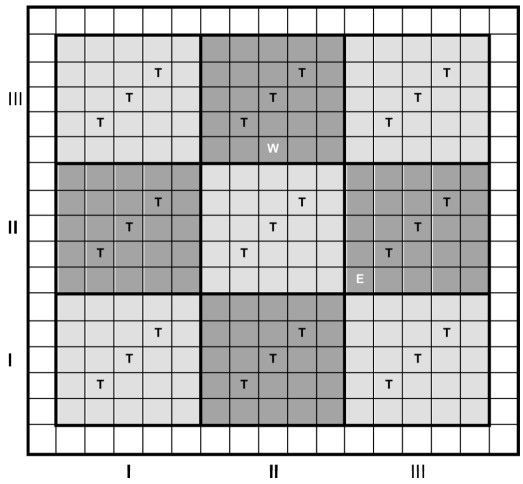

Figure 4.9: *Robots located at the triples ((II,III),(2,0),W) and ((III,II),(0,0),E).*

4.2.4 Batch Executions and Data Sets

We use the software (Section 4.2.2) to run a batch of `num_trials` world executions and generate a data set. The resulting data set is the second box on the left side of Figure 3.1. The world executions are numbered from 0 to $num\_trials - 1$. This number is called the *trial_number* and serves as the seed of the random number generator used by the agent function (Section 4.1.5). All world executions in the batch share the same initial world and parameters and differ only by the *trial_number*. The complete list of parameters along with the corresponding parameter values is given in Table 4.3. Table 5.2 shows the actual parameters that were used.

As a batch generates only a single data set, each data set includes a path for every robot in every world execution such that the robot moved through a positive number of regions in the world execution. Note that a robot will not have a path if it leaves the world before entering a region. When `num_trials` is 40,000, the cardinality $|PATHS|$ of a data set $PATHS$ for a small world with 25 robots and a large world with 50 robots will be at most 1 million and 2 million paths, respectively. Data sets record a number of world executions so that we must only setup and execute the worlds once. Once we have a dataset we can

create target matrices, path length histograms, and their corresponding visualizations, the heat maps and plots, without the need to setup and run another batch. They also provide more data than a single world execution, often combining thousands of world executions into a single data set for analysis.

4.3 Target Occupancy Matrices

We visualize data sets (Section 4.2.4) in several ways, including as sequences of target occupancy matrices (Section 4.3.1) and histograms of path lengths (Section 4.3.3). Since data sets are grouped by robot and not step they are not useful for quickly determining how regions are occupied in a step or the steps when the swarms leave regions. To quickly visualize these aspects of robotic behavior we create other data structures from the data sets.

The target occupancy matrices show the occupied regions in each step. They are the third and last data structure on the left side of Figure 3.1 as they are sufficient for observing batch executions (Section 4.2.4) and comparing them to predictions. Each target occupancy matrix represents a single step in a batch execution. This section details these matrices and how we produce them from a data set.

In addition to the target occupancy matrices, the histograms readily reveal how a swarm moves through a world, much like a wave, or through its regions individually.

4.3.1 Definition

A sequence of target occupancy matrices $O = \langle O^s \rangle$ aggregates a data set $PATHS$ by step s, with each cell O^s_p in each target occupancy matrix $O^s \in O$ representing the occupancy of a region at position p in the world at step s. Formally, each target occupancy matrix O^s provides the proportion O^s_p of path prefixes that end in the region at p in step s. This pro-

portion is out of the total number of robots in the batch execution ($num\_trials \cdot num\_robots$, the number of trials times the number of robots per trial) including those that never enter the world and those that leave. More formally,

$$O_p^s = \frac{\sum_{PATH \in PATHS} \sum_{l \in PATH} loc_{p,s}(s_{PATH}, l)}{num\_trials \cdot num\_robots} \tag{4.1}$$

where

$$loc_{p,step}(s, (p_i, q_i, r_i)) = \begin{cases} 1 & \text{if } p_i = p, s + i = step \\ 0 & \text{otherwise} \end{cases}$$

Here, l is the location at i, its index in the location sequence.

And to capture the proportion of robots that are outside of the world at step s, either because they exit or because they never enter:

$$OUT_s = 1 - \sum_{p \in REGIONS} O_p^s \tag{4.2}$$

since

$$OUT_s + \sum_p O_p^s = 1, \ 0 \leq s \leq max\_steps \tag{4.3}$$

Note that $OUT_0 = 1$ and $O_p^0 = 0$, for all p, since the swarm is outside of the regions at step 0.

4.3.2 Heat Maps

We visualize target occupancy matrices (Section 4.3.1) and prediction occupancy matrices (Section 4.6) with heat maps. The three types of heat maps for batch execution "West1" (in blue) and the corresponding prediction (in red) are shown in Figure 4.10. From Equation 4.1,

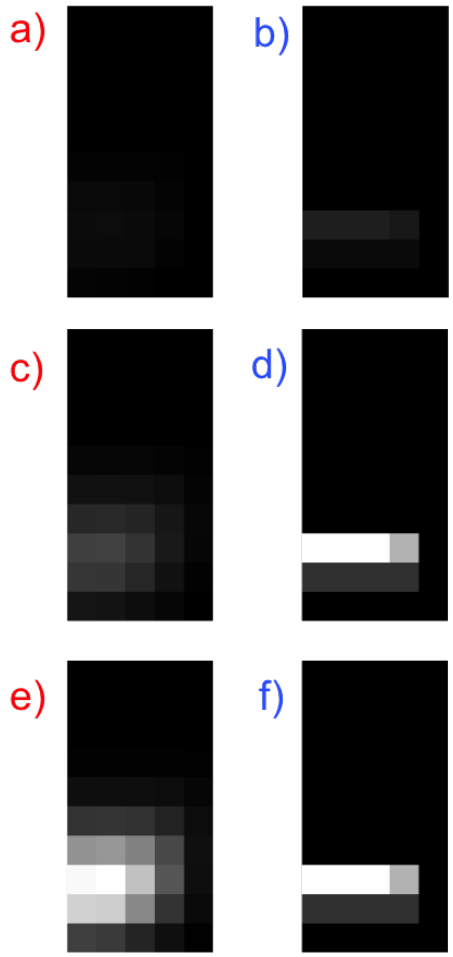

Figure 4.10: *Heat maps showing the occupancy and predictions of data set "West1" at step 30 that are (a,b) unadjusted, (c,d) adjusted together, and (e,f) adjusted individually.*

the values in the cells of the target occupancy matrices are normalized to be between 0.0 and 1.0 to show the proportion of the overall population of robots in each region to the overall population (the overall population does not decrease when robots leave). The shades of the regions in the heat maps are based on these proportions, using shades ranging from black for 0.0 to white for 1.0. Figure A.4 shows step 30 of the data set "West1" from Chapter 5. Compare the three regions with 4 robots, one region with 3 robots, and four regions with 1 robot in the world exection (Section A.4) with the target heat maps (Section 4.10) and see that the shades correspond to the proportions. In fact, the proportions in the target

occupancy matrices match those in the world execution exactly for this batch execution. This is because, as can be seen in Tables 5.1 and 5.2, the total number of trial numbers (and hence the number of random number generator seeds) for this data set is 1. Only one trial is required since the probabilities are all 0.0 or 1.0, making the data set deterministic. Then note that this is a small world, so the total number of robots for this data set is $25 = num\_robots = world\_width - 2 = 27 - 2$. The proportions in the cells of the target occupancy matrix shown in Figure 4.10(b) are $0.16 = \frac{4}{1 \cdot 25}$, $0.12 = \frac{3}{1 \cdot 25}$, and $0.04 = \frac{1}{1 \cdot 25}$, respectively.

The top-most proportions predicted in the regions of the heat map in Figure 4.10(a) are for the regions at (II, III), (I, III), and (I, II) and are 0.0517, 0.0509, and 0.0444, respectively. In Figure 4.10(a,b) we see heat maps for those values. The heat maps demonstrate how data sets or steps where robots are very spread out or many have left result in very dark shades if we do not make some sort of adjustment.

We adjust the shading in two ways: One way is to divide each matrix by the maximum value in the matrix, so that the values range between 0.0 and 1.0. The result is that the maximum value in the matrix appears white in the heat map and values close to the maximum value are brighter and more visible. Of course, this means that the shades are relative to the particular matrix and you can no longer compare the matrix to another of a different step or a different sequence (i.e. the target or prediction matrices). Figure 4.10(e,f) shows the heat maps adjusted this way. The adjustments for the values of the target occupancy matrix in Figure 4.10(f) above are $1.0 = \frac{0.1667}{0.1667}$, $0.75 = \frac{0.1250}{0.1667}$, and $0.25 = \frac{0.0417}{0.1667}$, respectively. The adjustments for the predicted occupancy matrix in Figure 4.10(e) are $1.0 = \frac{0.0517}{0.0517}$, $0.9845 = \frac{0.0509}{0.0517}$, and $0.8588 = \frac{0.0444}{0.0517}$, respectively.

The other way to adjust heat maps is to divide a target and prediction occupancy matrix of the same step by the maximum value from either of them. The values in the cells of two matrices adjusted using this method are relative to the step rather than the matrix. It is

possible to compare the values in the matrices to see how the proportions in the regions in one compare to the corresponding regions of the other. Figure 4.10(c,d) shows heat maps adjusted this way. The adjustments for the values of the target occupancy matrix in Figure 4.10(d) above are $1.0 = \frac{0.1667}{0.1667}$, $0.75 = \frac{0.1250}{0.1667}$, and $0.25 = \frac{0.0417}{0.1667}$, respectively. The adjustments for the prediction occupancy matrix in Figure 4.10(c) are $0.3101 = \frac{0.0517}{0.1667}$, $0.3053 = \frac{0.0509}{0.1667}$, and $0.2663 = \frac{0.0444}{0.1667}$, respectively. You'll notice how dark the colors are already for values as high as 0.26. Unless stated otherwise, it is safe to assume that a pair of heat maps are adjusted together and that they are comparable. We do not show unadjusted heat maps in the results.

We could also divide all values for a sequence of matrices or a pair of sequences by the maximum value of all of the matrices in the sequence(s) to make the steps more visible and comparable, but we do not do that for any heat maps shown in this paper.

While we use a sequence of target occupancy matrices to capture the distribution of robots at each step in a data set, we use histograms for each region to capture when robots left the regions. The histograms provide the distribution of the steps when the robots leave a region and, taken together, an abstraction of swarm progress through the world.

4.3.3 Path Length Histograms

Unlike the target occupancy matrices (Section 4.3.1) that aggregate a data set by step, histograms aggregate a data set by region. For any data set $PATH$, there is a histogram H_p for each region at p that describes the distribution of steps at which the robots leave the region. In determining if a robot exits a region at p in step s given a location l of a path $PATH$ in the data set $PATHS$ there are three cases:

- If we have reached the last location $l = (p_i, q_i, r_i)$, $p_i = p$ in the path but have not reached the last step of the world execution then the robot leaves the region by moving into the outer ring of squares and leaving the world.

- If we are in the region at p in step s and we are in region p' at step $s+1$, $p \neq p'$ then the robot exits the region by moving into another region.
- Otherwise, the robot does not leave the region either because the world execution ended or the robot remained in the region at p in the next step.

Definition

More formally,

$$H_p(s) = \sum_{PATH \in PATHS} \sum_{l \in PATH} exit_{p,s}(PATH, l) \tag{4.4}$$

where

$$exit_{p,s}(PATH, l_1) = \begin{cases} 1 & s_{PATH} + i = s,\ p_i = p,\ \neg \exists\, l_2 \,.\, l_2 \in PATH,\ s \neq \text{max\_steps} \\ 1 & s_{PATH} + i = s,\ p_i = p,\ \exists\, l_2 \,.\, l_2 \in PATH,\ p_j \neq p_i \\ 0 & \text{otherwise} \end{cases}$$

and

$$l_1 = (p_i, q_i, r_i),\ l_2 = (p_j, q_j, r_j),\ j = i + 1$$

H_p is the distribution of steps in which robots leave the region at p. Taken together, the histograms H_p show how robots advance through the world. This becomes more obvious when visualizing the histograms with box charts (Section 4.11). The count $\sum_s H_p(s)$ gives us an idea about which regions p hosted a higher concentration of robots over the course of the batch execution and can be viewed in heat maps (Section 4.13). We discuss the box charts and heat maps in the following sections.

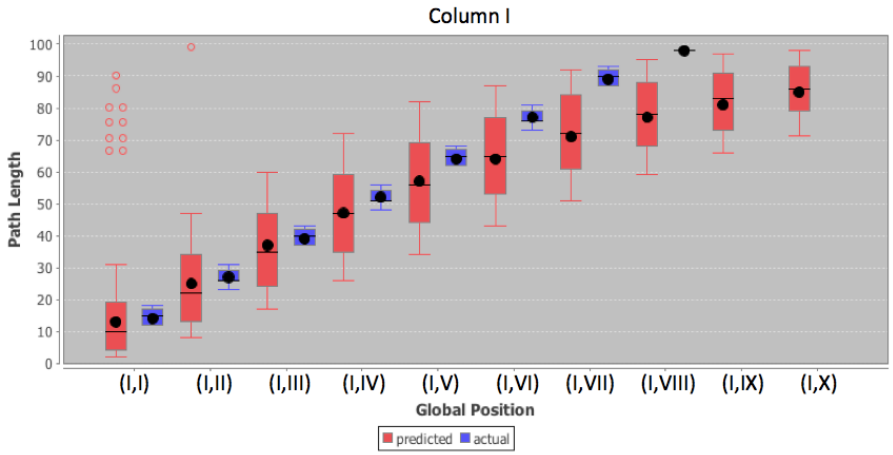

Figure 4.11: *A chart summarizing the histograms of column I for "West1" and graph "West1 B".*

Visualizing Histograms with Box Charts

We use two visualizations to visualize histograms. One is the box and whisker charts that we use to visualize path length distributions. The box chart comparing the histograms for column III of the batch execution "West1" and the corresponding evolutionary graph "West1 B" (shown in Figure 5.85) is shown in Figure 4.12. Notice how it reveals that the majority of exits from regions move steadily northward over time, suggesting that while some robots may remain behind, the swarm as a whole is making steady progress, the rate of which can be measured by taking the difference between the consecutive mean values. If the swarm slows down for any reason (or if it speeds up) you will be able to tell by the curve. The quartiles and optima give us an idea of how quickly the swarm moves through the regions. If they move through quickly the inter-quartile range will be short, but if they move through over a relatively long period of time it will be longer. In the example plot, the swarm exits each region within a few steps. It also takes the swarm a little while to move forward. Given 10 regions five squares tall, a swarm can reach the north edge within at least 50 steps, yet approaching step 70, the swarm only reaches the 6^{th} region row. Of

course, it turns out that the swarm is moving north and west and fewer and fewer robots are around in column III approaching the end of the batch execution.

The world at step 70 from the actual world execution is shown in Figure A.5. There we are not surprised to find that at step 70 there is only a single robot left in the III column and is heading into row VII. It is that single robot that moves through the region at (III, VII) that leads to that mean that we see for region (III, VII) in Figure 4.12. The rest of the robots are in columns I and II. The occupancy heat map for step 70 in Figure 5.12 is in agreement. Finally, in Figure 4.11 we see that plenty of robots make it into row VII in column I at around step 90.

You can also compare the box charts to the heat maps in Figure 4.10. The heat maps designated in blue show the swarm for this world execution at step 30. Most of the robots are in row III but a few are in row II. The plots show the robots leave row III around step 35 and that most robots leave row II by step 30. Furthermore, the heat maps in Figure 4.10 designated in red show the prediction spread thin around row III at step 30. Unlike the blue boxes, the red boxes show a long inter-quartile range and plenty of overlap. They show that the majority robots leave region (III, III) around step 30, but it overlaps with robots leaving regions II and III.

Visualizing Histograms with Heat Maps

Cumulative occupancy heat maps are the other way that we use the histograms to summarize what happened in a batch execution. The actual and predicted cumulative heat maps for batch execution "West1" are shown Figure 4.13. They show the total number of exits for each region and they suggest how far north robots travel and in what numbers in each region, column, and row. The cumulative occupancy heat maps are adjusted individually so that the highest value in the heat map shows as white.

In Chapter 5, we use heat maps to summarize target and prediction occupancy matrices

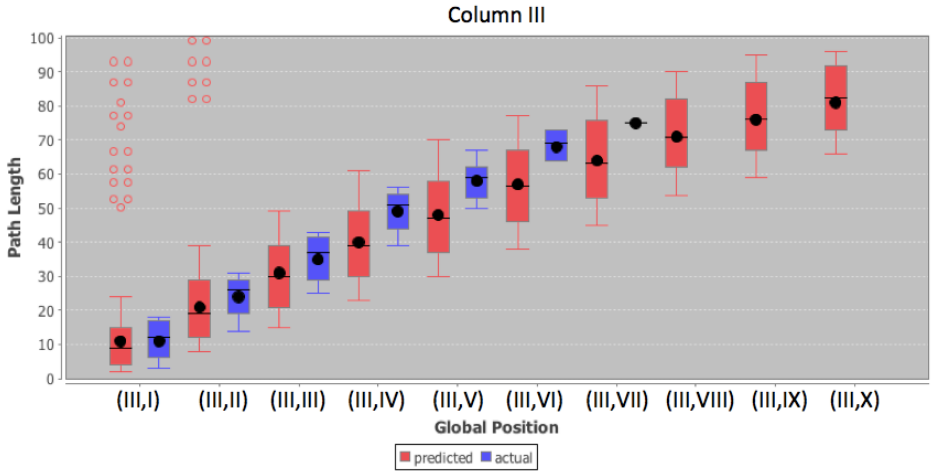

Figure 4.12: *A chart summarizing the histograms of column III for "West1" and graph "West1 B".*

the majority of the time, since we are most concerned with the ability of our graphs to predict the spatial distribution of a swarm over the uniform regions in each step. The histograms of data sets were frequently used to evolve graphs, something that is not a focus in this work.

4.4 Probabilistic Graphs

We use probabilistic graphs, such as the graph in Figure 4.14, to capture the swarm flow within and out of a region associated with a particular square pattern (Section 4.1.2) and set of parameter values (Section 4.2.4). We use them to generate a transition matrix (Section 4.5) and use the transition matrix to generate a prediction occupancy matrix (Section 4.6) for every step of any data set (Section 4.2.4) created using the same square pattern and parameters. The graphs are the first of the aforementioned data structures on the right side of Figure 3.1. In this section, we discuss the graphs and explain how we produce them from a data set. The example graph in Figure 4.14 is used to create the prediction occupancy matrices visualized in Figures 4.10, 4.13, 4.11, and 4.12 designated in red.

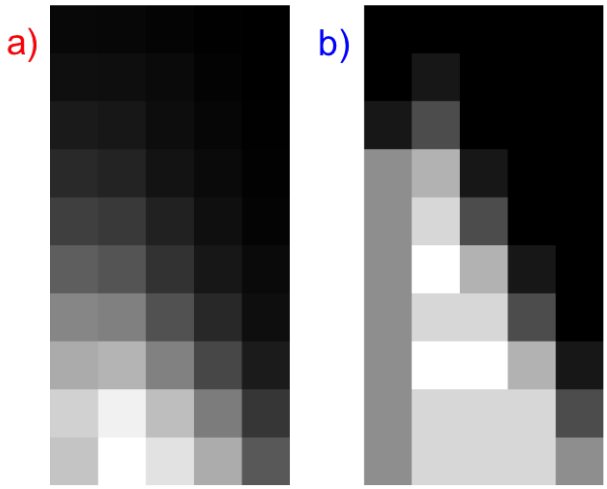

Figure 4.13: *Cumulative occupancy heat maps for batch execution "West1" and graph "West1 B".*

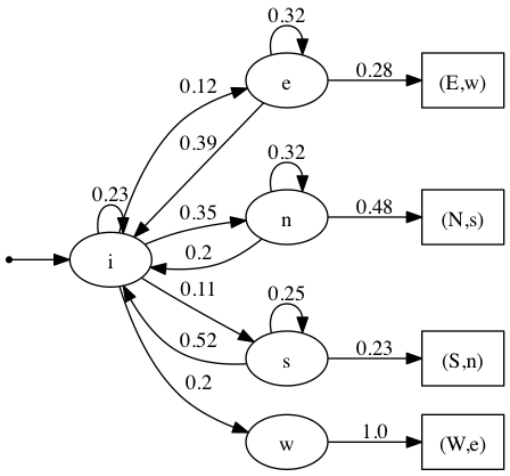

Figure 4.14: *Probabilistic graph "West1 B" for batch execution "West1".*

4.4.1 Graph Structure

Each graph captures how a swarm moves within and out of one or more regions. The associated square pattern (Section 4.1.2) interacts with itself and a swarm, given a set of parameter values (Section 4.2.4), to produce distinct behavior. The graphs summarize the rates at which robots occupy and leave a pattern in the associated regions and where they

go when they do.

The graph in Figure 4.14 has nine vertices. Strictly speaking it has only 5 vertices. The oval vertices are the internal vertices or "real vertices" and the proportion of the swarm considered to be inside these vertices is interpreted as being inside the region. The rectangular vertices, or "virtual vertices", point to a real vertex in a neighboring graph, if it exists. There must be at least one of these virtual vertices. There need not be a virtual vertex for every neighbor. There is no theoretical limit to the number of real vertices but the goal is to balance simplicity and predictive capability. They should be as informative as possible at a glance, but the necessary representational power may require a certain amount of complexity from the graph.

The value *gnode* (the "global vertex") in $(gnode, lnode)$ in the rectangular vertices determines whether the destination is the graph to the north, south, east or west and the value *lnode* determines which real vertex (or "local vertex") is the destination. In this graph, the vertices labeled "n", "s", "e", and "w" are closest to the north, south, east, and west neighbors, respectively. The vertex labeled "i" is the "inside" vertex and is not connected directly to a neighbor. Any proportion of robots that follow the edge into (E, w) end up in the region to the east in vertex "w". This means that robots move from "e" in the graph to "w" in the graph to the east in one step, if it exists. If located on the edge of the world so that no vertex is located at the destination of one of these rectangular vertices, the proportion that follows the edge is considered to leave the regions and is now outside of them. Figure 4.15 shows how a graph can be used to represent the swarm movement within and between regions. The graph is superimposed upon the 9 regions of a world to show how they are associated.

A glance at the example graph can tell us a few things. First, it is apparent that the swarm has a tendency to move north-west given the relatively large values of the edges leading to (N, s) and (W, e). It also shows the minimum number of steps that are necessary

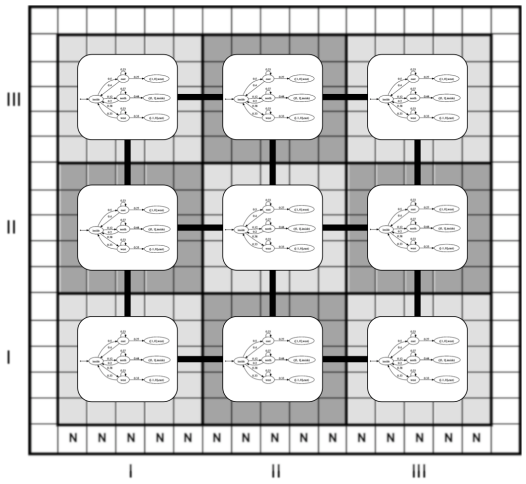

Figure 4.15: *A network of probabilistic graphs.*

for robots in a vertex of the graph to move into another graph. Only two steps are required for the graph in Figure 4.14, but quite a few steps are required for the hand crafted graph in Figure 5.13. The minimum number of steps is particularly important in these larger graphs. Compare this to the minimum of 5 steps required for robots to cross one 5x5 square region. Note that it takes 17 steps to cross a 5x5 square region diagonally since a robot can either turn or move forward one in a step and not both. The difference in the minimum number of steps between these two graphs can be explained by the cycles in the smaller graph. While it takes at most 12 steps for robots to move through the larger graph, they move through the smaller graph gradually with a proportion leaving in different steps. Nevertheless, the two graphs result in similar swarm distributions (see Figures 5.12 and 5.87) in their respective prediction occupancy matrices (Section 4.6) by predicting the highest occupancy in similar regions at each step.

4.4.2 Definition

The following formal definition of a graph will assist in our definition of transition (Section 4.5) occupancy matrices. First, let the function $g = graph(p)$, $g \in G$ be the graph

assigned to the region at p chosen from among the one to three graphs G created to capture local behavior in the regions. $|G|$ is often 1, but can be greater than 1 if mutually exclusive sets of regions show distinct behavior that cannot be captured by a single graph (Section 4.4.4). The definition begins in the usual way where

$$g = (V, E)$$

where V is the set of vertices of g and E is the set of edges. But we begin to customize our graphs by having $E = V^{'} \times V$. We account for virtual vertices by partitioning V into two disjoint sets, $V^{'}$ for real vertices and Γ for virtual vertices:

$$V = V^{'} \cup \Gamma$$

$$\emptyset = V' \cap \Gamma$$

The elements of the set Γ point to the local vertices in the graphs of neighboring regions:

$$\Gamma \subseteq \{(0,-1),(0,1),(-1,0),(1,0)\} \times \left(\bigcap_{g \in G} V_g^{'} \right)$$

While Γ is more constrained than necessary, it is sufficient to ensure that virtual vertices are valid when the associated neighbor exists.

Finally, $v^0 \in V^{'}$ is the start vertex of g and the mapping $P_{g \in G} : E_g \rightarrow \mathbb{R}$ maps the edges of a graph to probabilities.

4.4.3 Producing Probabilistic Graphs

Most of the graphs presented in the results in Chapter 5 are created by hand. This is possible when you have enough information about how robots move through a region. Whether or not it is possible may depend on the square pattern and the parameter values. Sometimes

it is enough to look at a square pattern. At other times it may be necessary to observe one or more world executions to better understand the movement. Either way it may be helpful to use automated methods to analyze the swarm flow in the square pattern.

We use all of the above methods to create our graphs. In particular, we present tables with statistics (specifically the mean and first and second quartiles) describing the proportions of the robots to exit the individual regions and the associated destinations for each relevant number of steps after entering. Table 5.4 is an example. The values in the table were computed automatically from the dataset for batch execution "West1" (Section 5.2) using software that we developed. The statistics were used to create graph "West1 A" in Figure 5.13. Notice how the graph has real vertices and edge probabilities that match both the exit timing and the exit proportions (using the means) from the table. Note that we were able to force the remaining robots to exit in the twelfth step and ignore the fact that statistics show one or more robots remaining in the regions after step 12. We are able to obtain nice results without worrying about the 5% that remained in the regions beyond step 12. We used the entire dataset to create "West1 A" because "West1" is determinstic and the corresponding dataset only contains up to 25 paths for 1 world execution of a small world. However, we generate statistics on subsets of a dataset for non-deterministic batch executions where there can be up to 1 million paths for 40,000 world executions of a small world. The graph "West4" (Figure 5.74) was created this way, by starting with the data for 1 world execution and doubling until we encountered diminishing returns.

In some cases it isn't enough to know when robots exit north, south, east, or west. This happens when robots have a strong tendency to follow one of a number of independent paths through the square pattern (Figure 5.70). In these cases, we use statistic tables to determine the swarm proportions originating in specific squares along the regions' borders and ending up in specific neighboring border squares (or simply remaining inside the region) n steps after entering.

There are two ways that we produce graphs from the statistics. One is to simply construct the graph to match the directions and timings of exits such as we did for "West1 A". But in the case of non-deterministic batch executions such as "West4" where we use multiple sample sizes, we must formalize the graph creation process to make it repeatable so that we can compare the graphs from the different sample sizes. In these cases we create a graph template that contains a number of unknown probabilities. While the graph template matches the timing and direction of exits as before, one or more key probabilities are left unspecified and must be filled in with values from the statistic tables corresponding to each sample size. The key probabilities are chosen by considering how robots move though the associated square pattern using flow graphics such as the one in Figure 5.70 for "West4". The flow in the figure reveals the probabilities used for each square. In the case of "West4", only the right turn probability is relevant since only the right turn probability is not 0 or 1. We chose the right turn probability as the only key probability for the graph template of "West4" for this reason.

We also produce graphs to represent a number of world regions using genetic algorithms by generating a best fit graph for one or more graph topologies. Graph topologies are graph skeletons of vertices and edges but without edge probabilities. To find a best fit graph, we evolve the edge probabilities. This method allows us to generate graphs quickly, even for world executions with relatively complex dynamics. The fitness function assigns higher fitness to graphs that produce histograms that are closer to the histograms of the data set. Then we choose the graph with optimal fitness from among the fittest graphs of the different topologies. We come up with a number of different topologies by hand rather than evolving them to decrease the search space and allow more control over structural bias.

Producing graphs is not the focus of this work but may be the focus of future research.

4.4.4 Using Multiple Graphs

While there are cases where swarm behavior in the regions is so similar that a single graph $g \in G$ is enough to represent them all ($|G| = 1$), there are also cases where two or three mutually exclusive sets of regions exhibit behavior that is distinct from the others, requiring the use of more than one graph ($|G| > 1$). In general, each region is assigned a graph $g = graph(p)$ that adaquately captures the swarm behavior within that region, where p is the position of the region. If the regions must be divided up into two or three sets, a graph must be created for each set, and each graph must be assigned to all of the regions of the corresponding set. When multiple graphs are used, it is usually the case that the regions in the left-most and right-most columns require their own graph and then one more graph is needed for the regions in the inside columns. In this case three separate graphs are required.

4.5 Transition Matrices

We use graphs (Section 4.4) to create *transition matrices* that we use for the transitions between occupancy matrices (Section 4.6.1). Transition matrices are the second data structure on the right side of Figure 3.1. By joining the graphs of the regions in two-dimensions we capture how the robots move globally, from region to region. The transition matrices connect graphs wherever the corresponding regions are connected in the world (see Figure 4.15). The generation of occupancy matrices from a transition matrix requires only matrix multiplication, making it very efficient. We can also grow or shrink transition matrices along either axis in order to make predictions about worlds of different dimensions. We describe how to construct them from probabilistic graphs in this section.

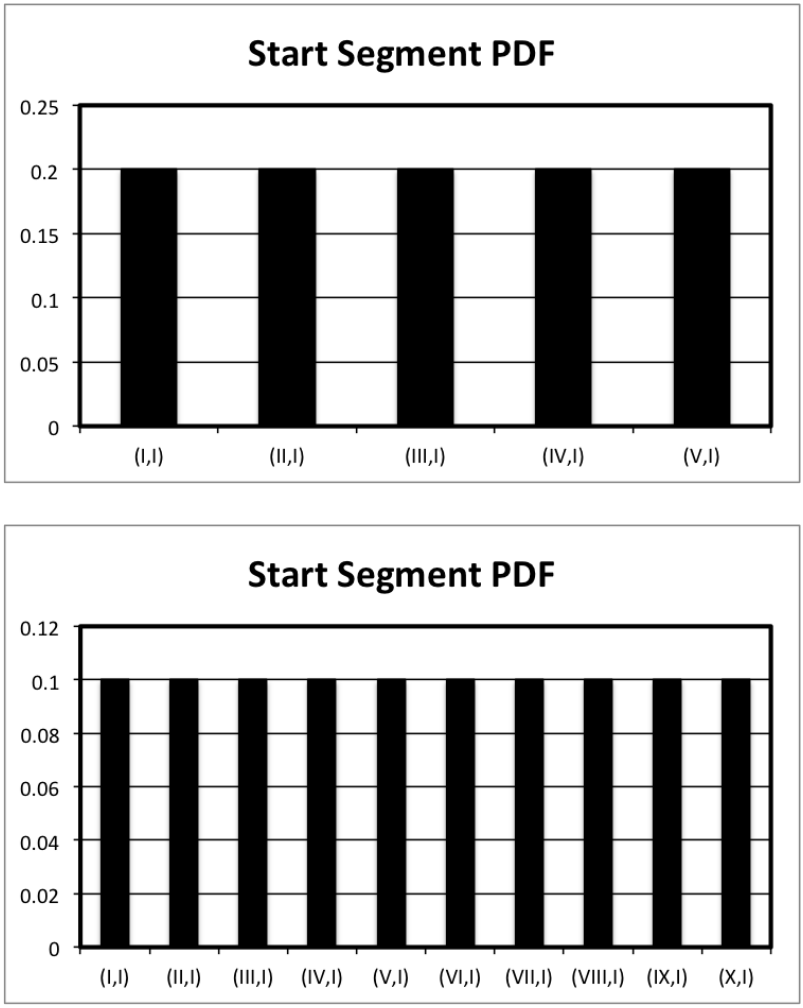

Figure 4.16: *The start region PDFs.*

4.5.1 Generating the Transition Matrix

A transition matrix T is generated automatically from one or more graphs $g \in G$. Graphs were defined in Section 4.4.2. We begin by defining a function *vmap* that assigns a unique non-negative integer to each real vertex $v \in V_g^{'}$ of every graph g. We use *vmap* when we define the function $index_p(v)$ that assigns every vertex $v \in V_g, g = graph(p)$ of every region at p an index into the transition matrix T.

$$vmap(g \in G, v \in V'_g) \in \mathbb{Z}^{\geq 0}$$

Similar to Section 4.4.2, we don't exactly define the real vertices, but we do need to impose a few constraints on *vmap*. Namely, that it assign a unique positive integer to every real vertex. We require that 0 is one of those integers and that consecutive integers are used for the rest. To begin, 0 is assigned to a vertex of every graph:

$$\forall\ g \in G\ .\ \exists\ v \in\ V'_g\ .\ vmap(g, v) = 0$$

Next, the integers must be assigned consecutively:

$$\forall\ g \in G\ .\ \forall\ v_1 \in V'_g\ .\ vmap(g, v_1) > 0 \Rightarrow\ \exists\ v_2 \in V'_g\ .\ vmap(g, v_2) = vmap(g, v_1) - 1$$

Finally, we require the integers assigned by *vmap* to be unique:

$$\forall\ g \in G\ .\ \forall\ v_1, v_2 \in V'_g\ .\ v_1 \neq v_2 \Rightarrow vmap(g, v_1) \neq vmap(g, v_2)$$

Next, we define the *rmap* function that assigns a unique consecutive integer to each region. This is important because we are representing two-dimensional coordinates in one dimension. We use *rmap* to define the function *order* that we use to determine the offset for the region at p when creating the indices $index_p(v)$ for $v \in V'_g, g = graph(p)$. This is necessary because the graphs g of G may have different numbers of real vertices, contributing differently to the offsets for the vertices of other regions. It also serves to provide a unique identifier that we use to refer to the regions in the context of assigning the indices.

$$rmap(p) = x_p * R\_HEIGHT + y_p$$

where $(x_p, y_p) = to\_arabic(p) - (1, 1)$ are the arabic numeral coordinates of the region at p starting from $(0, 0)$ and R_HEIGHT is the height of the world in regions.

Now we define the function $order(i)$ to be the number of real vertices in the graph of the region with index i.

$$order(i) = |V'_g|,\ g = graph(p) \wedge rmap(p) = i$$

Next, we define the function $offset(i)$ that defines the offset for the indices of a region.

$$offset(0) = 0$$

$$offset(i > 0) = order(i - 1) + offset(i - 1)$$

Then, we define the function $index_p(v)$ that will assign a unique index to each vertex of the graph for each region so that we can represent the probability of moving between any two vertices in the transition matrix:

$$index_p(v) = offset(i) + vmap(g, v),\ i = rmap(p) \wedge g = graph(p)$$

Lastly, we define the size of the matrix. The transition matrix must be able to accommodate a probability for any pair of vertices from the graphs of the regions and this requires it to have a row and column for each of the vertices. It must also have one additional row and column to account for robots leaving the world.

$$size = 1 + \sum_p (order \circ rmap)(p).$$

Now we can finally begin to build the transition matrix T:

$$T = zeros(size, size)$$

where the function *zeros* works like the function of the same name in SciLab or MatLab, producing a zero matrix with the dimensions provided.

Next we populate T using the edge probabilities given in the graphs of the regions. We must account for every edge, including those directed at virtual vertices. We start with the real vertices:

$$\forall\, p \in REGIONS\ .\ \forall\, e \in E_g\ .\ e = (v_1, v_2) \wedge v_1, v_2 \in V_g^{'} \Rightarrow T_{ji} = P_g(e),$$

where $g = graph(p)$, $index_p(v_1) = i$, and $index_p(v_2) = j$. P_g was defined in Section 4.4.2.

Finally, we define the transition probabilities for the virtual vertices:

$$\forall\, p, q \in REGIONS\ .\ \forall\, e \in E_g\ .\ e = (v, v_1) \wedge v_1 = (\delta, v') \in \Gamma \wedge q = p + \delta \Rightarrow T_{ji} = P_g(e),$$

where $v \in V_g^{'}$, $v^{'} \in V_h^{'}$, $g = graph(p)$, $h = graph(q)$, $index_p(v) = i$, and $index_q(v') = j$.

4.6 Prediction Occupancy Matrices

Now that we have transition matrices (Section 4.5) we can produce prediction occupancy matrices. Prediction occupancy matrices are the third and final datastructure on the right side of Figure 3.1. By visualizing prediction occupancy matrices with heat maps (Section 4.3.2) or even examining the matrices directly we can see regional occcupancy at a glance. They help us visualize the direction and rate of swarm movement. They also provide an alternative to observing many individual world executions. We create the prediction occupancy matrices for a batch execution (Section 4.2.4) using a sequence of intermediate occupancy matrices (Section 4.6.1) that we generate with a transition matrix and initial occupancy matrix. We covered the creation of transition matrices in Section 4.5.1. In this section, we

explain in detail how we produce prediction occupancy matrices using occupancy matrices.

4.6.1 The Intermediate Occupancy Matrices

Each *occupancy matrix* Q^s is an intermediate data structure storing the occupancy $Q^s_{index_p(v)}, 0$ of every real vertex $v \in V'_g, g = graph(p)$ of every region at p for a step s in a batch execution. While the predicted regional occupancy can be calculated from these matrices, it requires a summation over the occupancy of the vertices of each region (Section 4.6.2) and the matrices are one-dimensional so that they can be multiplied by a transition matrix. This all makes them unintuitive at a glance. In Section 4.6.2 we convert these one-dimensional occupancy matrices into two-dimensional prediction occupancy matrices $\hat{Q}^s$ that contain one cell $\hat{Q}^s_p$ for every region at p in a step s. Generating the occupancy matrix for step $s+1$ only requires us to multiply the transition matrix and the occupancy matrix for step s. We begin by building the occupancy matrix for step 0.

The Initial Occupancy Matrix

We begin to construct the initial occupancy matrix Q^0 with a call to *zeros*:

$$Q^0 = zeros(size, 1)$$

where the function *zeros* works like the function of the same name in SciLab or MatLab, producing a zero matrix with the dimensions provided. Next we use the probability distribution functions, shown in Figure 4.16, to populate this matrix:

$$\forall\, p \in REGIONS\ .\ Q^0_{i,0} = PDF(p), i = index_p(v^0_g) \wedge g = graph(p)$$

where v^0_g is the start vertex of the graph g.

Generating Occupancy Matrices

Once we have Q^s, we can generate Q^{s+1} with a single matrix multiplication:

$$Q^{s+1} = TQ^s$$

4.6.2 Generating the Prediction Occupancy Matrices

We have finally come to the point where we are ready to produce prediction occupancy matrices. Prediction occupancy matrices are directly comparable to target occupancy matrices (Section 4.3) and are readily visualized with heat maps (Section 4.3.2). To produce a prediction occupancy matrix $\hat{O}^s$ from an occupancy matrix Q^s, we have to sum the values for the real vertices $v \in V'_g,\ g = graph(p)$ for each region p. The resulting prediction occupancy matrix will have one cell for every region:

$$\forall\ s\ .\ \forall\ p\ .\ \hat{O}^s_p = \sum_{v \in V'_g} Q^s_{index_p(v),0}$$

4.6.3 Path Length Histograms

Prediction histograms $H_p(s)$ are created for a step s by traversing the edges of the graphs $g = graph(p)$ for all p starting in v^0_g with probability $PDF(p)$ where v^0_g is the start vertex for the graph $g = graph(p)$ and PDF is defined in Figure 4.16. Traversal continues until max_step is reached or the transitions lead to a virtual vertex with no corresponding graph (the robot leaves the regions). There are $num\_robots \cdot num\_trials$ such traversals. Any time a transition follows an edge between graphs we update the histogram, much like we did in Section 4.3.3. The algorithm is summarized in the pseudocode in Figure 4.17. Γ is the set of virtual vertices (Section 4.4.2). The functions `next_region` and `next_edge` make a random selection: `next_region` chooses p with probability $PDF(p)$ and `next_edge` chooses

```
for 1 to num_robots · num_trials
    s ← 0
    p ← next_region(PDF)
    v ← v_p^0
    while s < max_steps
        g_p ← graph(p)
        (V, E) ← g_p
        E' ← {e | e ∈ E, e = (v, _)}
        e ← next_edge(E')
        (v, v') ← e
        if v' ∈ Γ
            H_p(s) = H_p(s) + 1
            (δ, v) ← v'
            p ← p + δ
            if out_of_bounds(p)
                break
            end
        else
            v ← v'
        end
        s ← s + 1
    end
end
```

Figure 4.17: *Pseudocode for a prediction histogram.*

$e \in E'$ with probability $P_g(e)$ (note that $[\sum_p PDF(p)] = [\sum_{e \in E'} P_g(e)] = 1$). Finally, $H_p(s)$ is the histogram for the region at p in step s. Path length histograms were introduced in Section 4.3.3 where they were generated from the paths of a data set.

4.6.4 Excluded and Overestimate Heat Maps

We use excluded and overestimate heat maps to show how a prediction occupancy matrix underestimates or overestimates in the regions. An example can be seen in Figure C.2. Regions where both the target and prediction are 0.0 are colored green. Regions where the target and prediction are equal are white. Regions where the prediction is lower than the target are colored blue and higher errors are associated with darker shades. Regions where the prediction is higher than the target are colored red and higher errors are associated with

darker shades. A high prediction corresponds to the o (included) value (Section 4.7) for the region. A low prediction corresponds to the e (excluded) value (Section 4.7) for the region. The heat maps may be normalized by the highest error to make the error differences between regions more visible. Hence, the highest error will use the darkest shade whether it be red or blue (or both). If heat maps are normalized, it will be made explicit. Figure C.3 shows normalized heat maps that correspond to the unnormalized equivalent shown in Figure C.2. These heat maps are useful for determining how the individual regions contribute to the predictive value because the prediction value plots (Section 4.7) report aggregates and do not retain spatial properties concerning where the target or prediction robots actually are relative to each other or their position in the world.

4.6.5 Included Heat Maps

We use included heat maps to show the amount i (included) (Section 4.7) in every region. The included heat maps for graph "East" in steps 89-100 in Figure 5.9 is one example. While the exclusion and overestimate heat maps are useful for showing errors and the error types, it is not possible to tell whether both the target and prediction swarm occupy the same region. Sometimes it is useful to know not only that there is more or less prediction swarm in a region, but whether they both occupy the region, and which regions host the highest *included*. It can also be difficult to tell which regions are shared from the occupancy heat maps. Furthermore, included heat maps can be normalized to show the regions with the highest included and their comparison. Included heat maps are normalized by dividing all cells of the included matrix by the highest value in the matrix. Normalization results in the highest value in an included matrix being shaded white in the associated heat map and all values near the max value being be brighter and more visible.

4.7 Predictive Value

Our predictions take the form of prediction occupancy matrices (Section 4.6) that anticipate the regions where target swarms are expected to be and in what proportions. Qualitatively, a prediction occupancy matrix is judged by its ability to contain the target swarm in the corresponding target occupancy matrix (Section 4.3). A target occupancy matrix describes the proportion of the swarm in each region in a particular step. We judge predictions on a step-by-step basis. Our comparison of the prediction and target matrices is the final step in the process depicted in Figure 3.1.

We expect the prediction swarm to be "centered" on the most important regions, that is, the regions containing the highest proportion of robots in the target swarm. Prediction swarms in prediction occupancy matrices can be expected to be spread out a bit more than the target swarms in the target occupancy matrices, particularly if non-determinism or uncertainty provides for some amount of variation in behavior. On the other hand, if the prediction is spread too wide it loses value. We need to be able to anticipate where the majority of the swarm will be in a given step. We must be able to tell these relatively busy regions apart from the rest. Specifically, we expect a prediction to include most of the swarm and we expect most of our prediction to include the swarm.

We want to maximize the amount of the swarm that is included in our predictions and the amount of our predictions that include the swarm. Achieving a high value for the proportion of the actual swarm that is included by the prediction means that we correctly center on and include the majority of the swarm. Achieving a high value for the proportion of a prediction that includes the swarm means that our prediction is able to distinguish between regions that do contain a high proportion of robots and those that do not. A prediction that includes a high percentage of the swarm but where a large percentage of the prediction does not include any of the swarm is not a very useful prediction. For example,

we could predict that a swarm will be everywhere. We know that this prediction contains much of the swarm, but it isn't particularly helpful. Likewise, if most of our prediction includes the swarm but it only manages to include a small percentage of it then we learn very little.

Quantitatively, we measure the value of our predictions using two metrics: The first metric, *inclusion*, captures the ability of the prediction to include the swarm. The second metric, *discriminatory value*, captures the ability of the prediction to discriminate between occupied and unoccupied regions. We also show the exclusion, which reveals the proportion of the swarm that the prediction misses, a value that we want to minimize. Combined, these metrics provide an objective measure of how close a prediction occupancy matrix is to a target occupancy matrix. To compute the value of these metrics, we begin by computing the following quantities: (1) the amount of the swarm in each region that is included by the prediction, (2) the amount of the swarm in each region that is excluded by the prediction, and (3) the amount of the prediction in each region that does not contain any of the swarm. Figure 4.18 depicts a region from a prediction occupancy matrix and the corresponding region in a target occupancy matrix in two different cases. In one case the predicted proportion is at least as high as the target proportion and in the other the target proportion is at least as high as the predicted proportion.

More formally, for a given step s, target occupancy matrix O^s, prediction occupancy matrix $\hat{O}^s$, and global position p, the proportion that is included by the prediction, from Figure 4.18, is $i = min\{O^s_p, \hat{O}^s_p\}$. If the actual occupancy is higher than the predicted occupancy ($O^s_p \geq \hat{O}^s_p$) then the amount of the swarm that is excluded is $e = O^s_p - \hat{O}^s_p$. Otherwise the proportion of the prediction that did not contain any of the swarm is $o = \hat{O}^s_p - O^s_p$. Note that for a given p, either o is positive or e is, but never both. In other words, either the prediction was too high or too low, not both. Of course, it could be exact ($o = e = 0$). Also, i can be 0. In general, $0 \leq i + o + e \leq 1$.

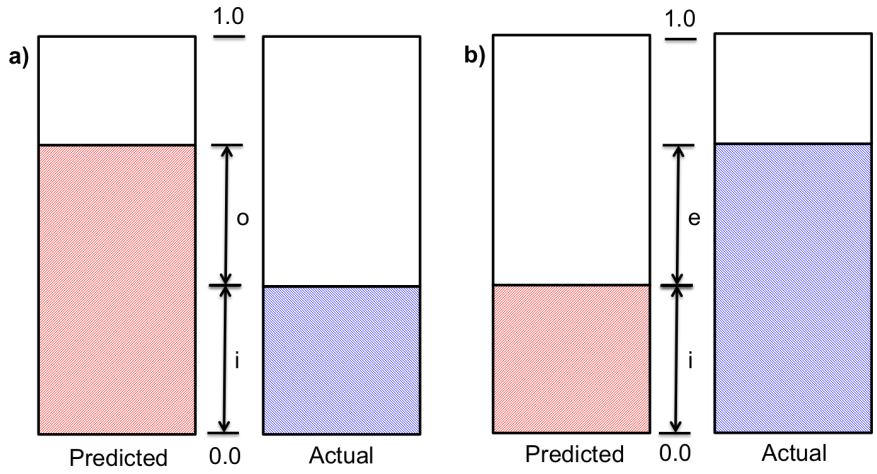

Figure 4.18: *Calculating the included, excluded, and overestimate for a region.*

The included i, excluded e, and overestimate o for a step s is:

$$included_s = \sum_p min\{O_p^s, \hat{O}_p^s\} \tag{4.5}$$

$$excluded_s = \sum_p excl(O_p^s, \hat{O}_p^s) \tag{4.6}$$

$$overestimate_s = \sum_p over(O_p^s, \hat{O}_p^s) \tag{4.7}$$

where

$$excl(t,q) = \begin{cases} t - q & t > q \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

and

$$over(t,q) = \begin{cases} q - t & q > t \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

Then,

$$inclusion_s = \frac{included_s}{included_s + excluded_s}, \tag{4.10}$$

$$discriminatory\_value_s = \frac{included_s}{included_s + overestimate_s}, \tag{4.11}$$

and

$$exclusion_s = \frac{excluded_s}{included_s + excluded_s}. \tag{4.12}$$

Note that, for any step s,

$$included_s + excluded_s = \sum_p O_p^s \tag{4.13}$$

and

$$included_s + overestimate_s = \sum_p \hat{O}_p^s. \tag{4.14}$$

Finally, note that

$$exclusion_s = 1 - inclusion_s. \tag{4.15}$$

4.7.1 Sensitivity and Bias

To aid in our assessment of predictive value, we include sensitivity and bias in our list of prediction quantification metrics. An example sensitivity and bias plot that plots the sensitivity and bias in each step for the graph "East" can be found in Figure 5.11. Sensitivity, A, attempts to measure how well the values in a prediction occupancy matrix predict the

values in target occupancy matrix. It does so using the hit rate H and the false-alarm rate F. The hit rate, H, is the proportion of the target swarm that overlaps with a prediction swarm. The hit rate increases as more of the target swarm overlaps with a prediction swarm. The false alarm rate, F, is the proportion of a prediction swarm that does not overlap with the target swarm. The false alarm rate increases as the proportion of the prediction swarm that overlaps with the target swarm decreases. The sensitivity tends to be high when the hit rate is high relative to the false alarm rate. Bias, b, attempts to measure the tendency of a prediction occupancy matrix to overestimate or underestimate the distribution of a target swarm in the regions. The bias is a function of H and F that is described in Figure 4.19. The sensitivity and bias are plotted together. We discuss the limitations of H and F and our assumptions in Section 4.7.1.

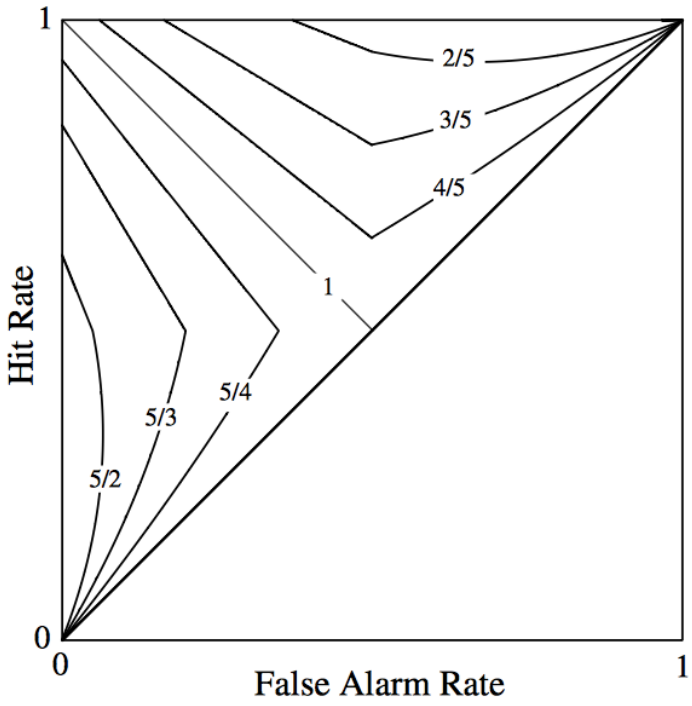

Figure 4.19: *The isopleths for the bias measure b.*[1]

[1] Zhang and Mueller (2005)

We use the non-parametric measures A and b (Zhang and Mueller 2005) to estimate sensitivity and bias, respectively. By non-parametric, it is meant that they do not rely on the underlying signal and noise distributions. They are defined as follows:

$$A = \begin{cases} \frac{3}{4} + \frac{H-F}{4} - F(1-H) & (F \leq 0.5 \leq H) \\ \frac{3}{4} + \frac{H-F}{4} - \frac{F}{4H} & (F \leq H < 0.5) \\ \frac{3}{4} + \frac{H-F}{4} - \frac{1-H}{4(1-F)} & (0.5 < F \leq H) \end{cases} \tag{4.16}$$

$$b = \begin{cases} \frac{5-4H}{1+4F} & (F \leq 0.5 \leq H) \\ \frac{H^2+H}{H^2+F} & (F < H < 0.5) \\ \frac{(1-F)^2+(1-H)}{(1-F)^2+(1-F)} & (0.5 < F < H) \end{cases} \tag{4.17}$$

where H is the hit rate (Equation 4.18) and F is the false-alarm rate (Equation 4.19). A ranges between 0.5 and 1.0. $A = 1.0$ is interpreted as perfect sensitivity and $A = 0.5$ is interpreted as a complete lack of sensitivity. Bias is considered to not be present if $b = 1$. A value $b < 1$ indicates a tendency to overestimate in the regions and a value $b > 1$ indicates a tendency to underestimate in the regions. Note that neither A nor b are defined if $H < F$. When the hit rate H is lower than the false alarm rate F it is considered to be a sampling error or response confusion (responding yes when intending to respond no and vice versa). When $H < F$, which happens when graphs do not match the occupancy in the regions well, we do not plot a point in the sensitivity and bias plot.

We define the hit rate H and the false-alarm rate F as follows:

$$H = \frac{included}{included + excluded} \tag{4.18}$$

and

$$F = \frac{overestimate}{included + overestimate}, \tag{4.19}$$

where *included*, *excluded*, and *overestimate* are defined in Section 4.7.

Arguments For and Against Using A and b

It is important to note that the comparison of our prediction occupancy matrices does not represent a binary classification problem. Rather, we are comparing the quantities in the cells of a prediction occupancy matrix and a target occupancy matrix. Rather than presenting signal trials and noise trials and accepting yes or no responses for them, we present a target occupancy matrix with a number of cells, each one including a number between 0.0 and 1.0 that represents the distribution of the target swarm in the associated region in the associated step. Then we must determine how closely matched the values in the cells of the associated prediction occupancy matrix are to those in the target occupancy matrix. As such, it is not clear what the underlying signal and noise trials are. It is tempting to think of the amount of prediction swarm in a region as yes responses and the amount of target swarm in the regions as the signal trials. Then perhaps by subtracting the amount of prediction swarm from 1 we obtain the no responses for the region and by subtracting 1 from the amount of target swarm we obtain the noise trials for the region. But it is not clear what an individual signal trial or signal is since we are working with continuous values. Even so, there does not seem to be a point where we present noise trials and signal trials. If we do, we do not expect a yes or no response, and there is not the presence or absence of a signal. While rankings can be associated with responses, the rankings still only represent a subjective uncertainty about the whether or not the signal is or is not present. The decision variable could be based somehow on the prediction occupancy matrix, but the decision variable values are typically scalars. So it is also not clear what the criteria would look like. It is also not clear what the underlying signal and noise distributions are. We use the metrics under the assumption that choosing a hit rate H and false-alarm rate F that make sense is sufficient to achieve valid results from A and b. We also use non-parametric

meausures of sensitivity and bias to avoid dependence on the underlying distributions. In what follows, we discuss the trade-offs for or our choice of H and F.

The hit-rate H in Equation 4.18 seems to make sense because it is 1 when the prediction swarm includes the entire target swarm and 0 when it misses the entire target swarm. Furthermore, H increases linearly with the proportion of the included target swarm and H does not depend on *overestimate*. So the prediction swarm can be much larger than the target swarm, but including the entire target swarm will still yield an H of 1. From Equation 4.18, *included* is the "number of hits" and $included + excluded$ is the "number of signal trials". So if thinking in terms of the prediction swarm being the yes responses, the compliment of the prediction swarm as being the no responses, and the target swarm being the signal trials, then the hits are the number of times that our predictions respond yes to a signal trial. But since we are using quantities and since we don't have much information about underlying signal or noise trials, we go with our assumption that if H and F make sense, then so will A and b. One potential caveat is that H depends on *included* and *excluded* which are summations of i and e over the prediction and target matrices. The *included* is our estimate of true positives and the *excluded* is our estimate of false negatives. But these are estimates taken over continuous values. Also, only *excluded* or *overestimate* can be positive at the same time for any region. This means that, if we consider the target swarm in a region to be the "signal trials" and the prediction swarm in a region to be the "responses", then there cannot be misses and false alarms for the same region, it will be one or the other depending on whether the target occupancy or prediction occupancy is higher in the region. It also means that the signal trials are all hits in a region as long as the amount of responses are at least the amount of signal trials. Hence, H merges the responses and signal trials. Another caveat is that H can be written as a function of F since they are both a function of *inclusion*. But since it makes sense for H to depend on *inclusion* and *exclusion*, we defer that argument until our discussion of F.

As for the false-alarm rate F, it is 1 when the prediction swarm completely misses the target swarm and 0 when all of the prediction swarm includes the target swarm. Also, F increases with proportion of the prediction swarm that does not include the target swarm. However, what is not clear is whether it makes sense for $included + overestimate$ to be the "number of noise trials". It seems suspect that the number of yes responses (the prediction swarm) is the same as the number of noise trials (the prediction swarm again). For example, how would you answer yes to a noise trial? It also seems suspect that the "number of noise trials" depends on H, the number of hits, and the number of false alarms ($F = \frac{included}{included+overestimate}$), where $included$ is our "number of hits" and $overestimate$ is our "number of false alarms". Once again, since F seems to lack a firm foundation in signal detection theory, we assume that if H and F make sense in context of our problem then we can obtain useful values from A and b.

In practice, the sensitivity and bias seem to be useful when using F (Chapter 5). The sensitivity tends to look much like the inclusion, but the bias seems to be a legitimate measure of overestimation or underestimation in the regions.

However, to deal with the issues presented by F, we could define F' instead:

$$F' = \frac{overestimate}{\overline{target}} \tag{4.20}$$

where

$$\overline{target_s} = \begin{cases} \sum_p 1 - O_p^s & (O_p^s + \hat{O}_p^s > 0) \\ 0 & (otherwise) \end{cases} \tag{4.21}$$

The difference between F' and F is the "number of noise trails", which we define to be $\overline{target}$. So the "number of noise trials" is 1 - "number of signal trials" in a region or the "amount of responses" that should be no. If the prediction is low, it answers no to some

amount of signal trials and if the prediction is high it answers yes to some number of noise trails. This value does not depend on either *inclusion* or *overestimate*. Also, H and F' are independent. It also does not depend on the size of the prediction swarm within the world. It seems to make more sense for $\overline{target}$ to be the "number of noise trials" than for $included + overestimate$ to be the "number of noise trials" for these reasons. However, F' has its own problems. We only include regions that are occupied in an attempt to prevent the "number of noise trials" from growing too large since $0 \leq o \leq 1$ but even when only counting the number of occupied regions $0 \leq \overline{target} \leq num\_regions$, where $num\_regions$ is the number of occupied regions. So now we have the problem where, with few exceptions, the number of false alarms can never equal the "number of noise trials" even if the entire prediction swarm misses the target swarm. Also, the number of noise trials jumps around as the target swarm moves into different numbers of regions or leaves the world. And this might be okay, but due to the incredibly small F' the sensitivity looks a lot like the inclusion (or the hit rate) in shape and the bias looks a lot like the exclusion in shape, only the sensitivity and bias are on different scales. As far as scale, A stays very close to 1. As for b, it can reach values well above 1 such as 9 and 12. In fact, F' is so low that it has very little effect on A or b. It is no surprise then that both A and b vary very strongly with H. Even changing number of occupied regions doesn't seem to effect it. To show that A and b vary strongly with H, we define A^* and b^* as

$$A^* = \frac{3}{4} + \frac{H}{4} \tag{4.22}$$

and

$$b^* = 5 - 4H \tag{4.23}$$

These functions A^* and b^* are only A and b with $F = 0$, respectively. If A and b truly

vary directly with H most of the time then they will be close to A^* and b^* most of the time. Figure D.1 shows that this is the case for graph "East". We include A and b along with A^* and b^* for all graphs in Appendix D for the interested reader. Note that every now and then H can drop low enough that F' begins to have an influence. But these are for very low values of inclusion where A and b are not even defined with F since in these cases $H < F$. The plot of A and b using F' for batch execution "West1 B" in Figure D.24 is one example. Note that the predictive value in Figure 5.90 and the sensitivity and bias (using F) in Figure 5.91 are very poor and undefined toward the end, respectively. The fact that A and b calculated with F' vary directly with H in almost all cicumstances makes them unlikely to be valid and not particulary useful.

Note that we could also define an F'' where we use all regions instead of only occupied regions that would not vary with the number of occupied regions. F'' would also not depend on *included* or *overestimate*. F'' would not be dependent on H. But F'' would be even smaller than F and A and b would again depend directly on H.

We assume that a meaningful H and F, along with the non-parametric measures A and B will provied meaningful values for the sensitivity and bias. The alternative false-alarm rate F' has very low values generally, making it a bad false-alarm rate. The values of H, F', and F in batch execution "East" for step 37 are 0.8481, 0.0087, and 0.1279, respectively. That is, $H = \frac{848}{1000}$, $F' = \frac{9}{1000}$, and $F = \frac{128}{1000}$, respectively. The inclusion and discriminatory value where

$$inclusion = 0.8481 = \frac{0.6106}{0.7200} = \frac{0.6106}{0.6106 + 0.1094} = \frac{included}{included + excluded}$$

and

$$discriminatory\_value = 0.8721 = \frac{0.6106}{0.7001} = \frac{0.6106}{0.6106 + 0.0895} = \frac{included}{included + overestimate}.$$

In this case, while F seems to make sense, suggesting that the false alarms were about 10% of the total prediction, F' is much too low to make sense. A value of $\frac{9}{1000}$ suggests that there were only 9 false alarms out of 1000 noise trials. But it would make far more sense to have $\frac{128}{100}$ since that appears to make more sense in the context of the inclusion and discriminatory value. Of course, $H = inclusion$, so it is no surprise that they are the same. This suggests that H and F are a suitable hit rate and false alarm rate. We only use F in the results. We discuss the sensitivity and bias for each graph in Chaper 5.

4.7.2 Supplemental Predictive Value Plots

Appendix C contains a number of supplemental predictive value plots. The predictive value supplement plots include the total prediction, total occupancy, included, excluded, and overestimate. The total prediction is the proportion of the prediction swarm in the world. The total occupancy is the proportion of the target swarm in the world. They also include the number of occupied regions, the number of regions occupied by both the target swarm and the prediction swarm (shared), the number of regions occupied by the target swarm, the number of regions occupied by the prediction swarm, the number of regions occupied exclusively by the target swarm, and the number of regions occupied exclusively by the prediction swarm. There is also the regional predictive value which includes the inclusion, exclusion, and discriminatory value computed for each region and then averaged. Finally, there are the regional predictive value supplement plots that include the total prediction, total occupancy, included, excluded, and overestimate averaged over the regions. Example plots can be found in Figure C.1. These supplemental plots can used to make sense of the

inclusion, exclusion, discriminatory value. Furthermore, the regional occupancy counts and regional averages provide a picture of how spread out the prediction swarm and target swarm are and the value of predictions on a region-by-region basis. Note that the average values may be lower than their counterparts since the values must be summed together to produce the latter. In addition, quite a few zeros can bring down the average inclusion, exclusion, and discriminatory value. For example, the inclusion may be 0 even if the overall inclusion is 1 since regions with only overestimate will have an inclusion of 0. Nevertheless, the average values remain informative, as they provide a picture of spread as well as predictive value and may be used to clarify features in the other plots.

4.7.3 Supplemental Sensitivity and Bias Plots

In addition to supplement plots for predictive value, we include supplement plots for the sensitivity and bias in Appendix D. The sensitivity and bias supplement plots include the hit rate, false alarm rate, number of hits, number of false alarms, number of signal trials, and number of noise trials for the sensitivity and bias that use F and the sensitivity and bias that use F' (Section 4.7.1). Example plots can be found in Figure D.2. These supplemental plots can used to make sense of the sensitivity and bias.

We present the results in the next chapter.

Chapter 5

Results

We present the results of using probabilistic graphs (Section 4.4) to predict the direction and rate of swarm movement in batch executions (Section 4.2.4). Table 5.1 outlines the batch executions along with a pointer to the parameter values used in Table 5.2.

5.1 Predicting an Eastward Path

We begin to demonstrate how graphs can be used to predict the direction and rate of swarm movement with a batch execution where the swarm moves north-east. The world (Section 4.1) "East" is constructed from the square pattern (Section 4.1.2) "East" shown in Figure 4.2(a). The initial world is shown in Figure A.1 in Appendix A along with the initial worlds for the other batch executions.

This world is free from target interference since the square pattern contains only obstacles and no targets (Section 4.1.1). Since there are no targets, `heat_range` and `sonar_range` (Section 4.1.4) are irrelevant (assuming `sonar_range` is at least 1) and `heat_range` and `sonar_range` for this batch execution are shown as N/A in the parameter values tables (Table 5.1 and Table 5.2). Robots only use their sonar to see beyond adjacent squares when

Table 5.1: *The 10 batch executions.*

Number	Name	Pattern	World Dimensions	Parameter Values
1	East	East	Small	1
2	West1	West1	Small	2
3	West2	West2	Small	2
4	West3	West1	Small	1
5	West4	West1	Small	3
7	West1 Large	West1	Large	2
8	West2 Large	West2	Large	2
9	West3 Large	West1	Large	1
10	West4 Large	West1	Large	3

Table 5.2: *Parameter values used for the batch executions.*

#	Trial Numbers	Steps	Heat Range	Sonar Range	Phototaxis Probability	Lateral Phototaxis Probability	Follow Probability	Right Turn Probability
1	N/A	100	N/A	N/A	1.0	1.0	1.0	1.0
2	N/A	100	3	3	1.0	1.0	1.0	0.0
3	$[0, 40K)$	100	N/A	N/A	1.0	1.0	1.0	0.5

in the presence of heat and without targets there is no heat. In the absence of heat, the robots move forward if they are not in collision with an object. If there were targets, robots would be distracted and forced to turn even if they were not in collision. They could even turn toward an obstacle or other robot, missing the target emitting the heat. Hence, there are no targets in batch execution "East" to prevent the swarm from moving easily through the world.

Robots are also relatively unaffected by obstacles since `lpp`, `fp`, and `rtp` are all equal to 1.0. Since `lpp` is 1.0, according to the rules of the agent function (Section 4.1.5), robots will always turn toward the north if they are moving east and are unobstructed on their northern side. In addition, since `fp` is 1.0, if facing east (north), in collision, and unobstructed on the south (west) side, they will always turn toward north (east). Since `rtp` is 1.0, robots always turn east when facing north and in a collision that does not trigger a follow move. Since

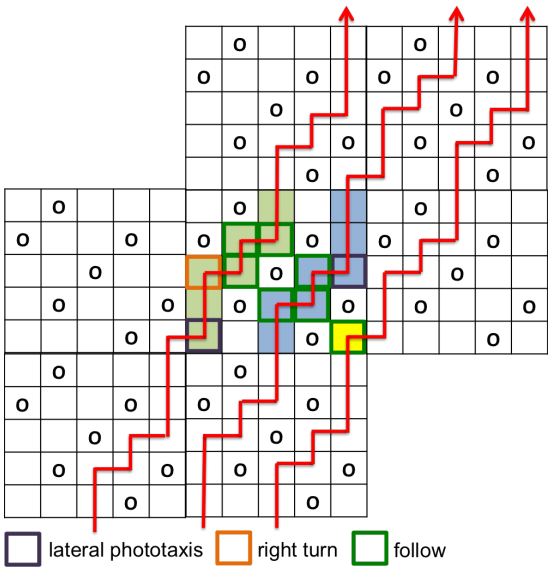

Figure 5.1: *The local behavior of batch execution "East".*

`lpp` and `pp` are 1.0 robots will always turn north, if they are not already facing north and are able. In batch execution "East" in particular, if heading east or north, all follow moves and turns resulting from collisions cause robots to turn north or east, respectively. This all translates into a strong propensity for directed movement in the north-east direction. Figure 5.1 depicts this rapid movement north and east (assuming no other robots get in the way) through the region in the center. For example, in Figure 5.1, a robot in the square with coordinates (Section 4.1.3) $(0, 0)$ facing east will always turn to face north since `lpp` is 1. A robot facing north in $(0, 0)$ or $(0, 1)$ will always move north into $(0, 1)$ or $(0, 2)$, respectively. A robot facing north in $(0, 2)$ will always turn to face east since `rtp` is 1. A robot facing east in $(0, 2)$ will always move forward 1 into $(1, 2)$. A robot facing east in $(1, 2)$ will always turn to face north since `fp` is 1. Assuming a robot does not collide with another robot, it will move through the region as depicted in Figure 5.1. All collisions occur in row I at the beginning of the batch execution. Once out of row I, robots reach the last square 13 steps after starting on the green path, 11 steps after starting the blue path, and 2 steps

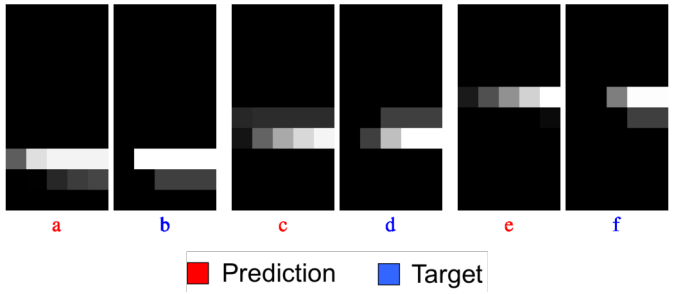

Figure 5.2: *Predicted occupancy of graph "East" at steps 30, 50, and 70.*

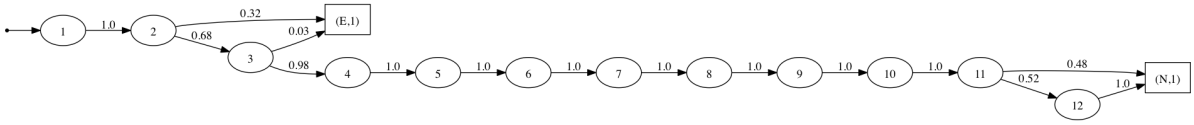

Figure 5.3: *Graph "East" for batch execution "East".*

after starting the yellow path. These paths have these long lengths because robots can only rotate or move forward in a turn and not both.

The target occupancy matrices for steps 30, 50, and 70 are shown in the heat maps (Section 4.3.2) in Figure 5.2(b,d,f) designated with the blue letters. They show the proportion of the target swarm in each region in each of those steps in the batch execution. To see how the target occupancy matrices (Section 4.3) represent the batch executions, compare the heat map for the target occupancy matrix at step 70 with step 70 of the world execution (Section 4.2.1) in Figure A.2. Notice that there are three shades in the heat map and the brightest shade (white) corresponds to the two regions with 4 robots in the world execution, the second brightest with the one region with two robots, and the darkest with the two regions with one robot. Note that this target occupancy matrix corresponds directly to this world execution because the batch execution is deterministic. If the batch execution had been non-deterministic (if any of the probabilities had been in $(0, 1)$) the distribution in the target occupancy matrix would be an average over 40,000 separate world executions and would not be an exact match.

Table 5.3: *Exit analysis of batch execution "East".*

	Remain			East			North		
Step	Q1	Mean	Q3	Q1	Mean	Q3	Q1	Mean	Q3
1	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.5179	0.6667	1.0	0.0	0.3209	0.4821	0.0	0.0	0.0
3	1.0	1.0	1.0	0.0	0.025	0.0	0.0	0.0	0.0
4	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
9	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
10	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.4	0.5179	0.6	0.0	0.0	0.0	0.5	0.4821	0.6
12	0.0	0.0804	0.0	0.0	0.0	0.0	1.0	0.9196	1.0

We are able to capture the north-east direction and rate in the regions of the batch execution with the graph shown in Figure 5.3. We use the table of statistics (Section 4.4.3) in Table 5.3 to produce the graph. We track the robots that enter each of the 50 regions to determine the proportion that either remain or leave to the north, south, east, or west after remaining in the region for s steps, $s \geq 1$. The statistics summarize the proportions for all of the regions. We use the averages to create the graph and the averages do not capture all behavior. Note that the large inter-quartile and inter-optima ranges may indicate that, by using the averages, we are missing behavior that may deserve separate treatment. Also the last average is not 1, meaning that robots sometimes remain in the region after 12 steps. These exceptions are from the swarm flow in row I when the robots first enter the world and do not represent the behavior in rows II to X. We discuss similar exceptional behavior in more depth and present a few different ways to deal with it in Sections 5.5, 5.7, and 5.8. The graph captures the behavior of the majority of the swarm despite not accounting for the exceptional behavior.

The virtual vertices in the graph (p49), that is, vertices that represent real vertices in

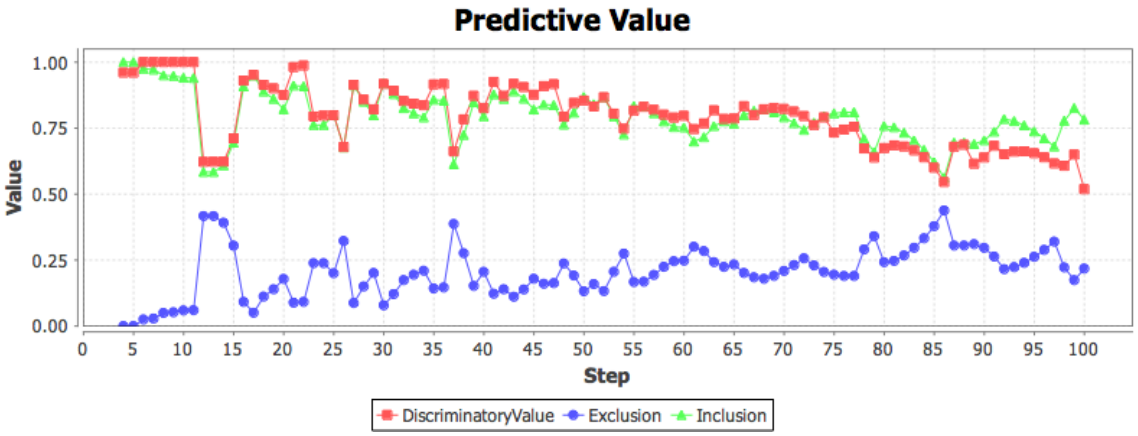

Figure 5.4: *Predictive value for graph "East".*

the graphs of neighboring regions, reveal that robots in the prediction swarm only move east via vertex "E" and north via vertex "N" and never west or south. It also shows the proportion of the prediction swarm that will exit a region into the east and north neighbor after remaining in the region for a specific number of steps. For example, it would take a robot at least 3 steps to reach "E" and at least 12 steps to reach "N". In another example, the graph shows that in step 3 32% of the robots will move to the virtual vertex "E", moving into the region to the east, if it exists. If no such region exists, 32% of the robots will leave the world. The other 68% will move to real vertex "3" and remain in the associated region until exiting north via "N" after a total of 11 or 12 steps. The graph contains 12 vertices and 15 edges. It may be possible to capture the behavior of this batch execution using a smaller (or larger) graph, yet the graph is sufficient to demonstrate that we can obtain a very accurate result using these graphs. Using a simpler graph may also come at the cost of predictive value, presenting a trade-off. The graph, as is, is easy to understand and so there may not be much of a trade-off after all. Optimizing the graphs is left to future work. The prediction occupancy matrices (Section 4.6) shown in the heat maps in Figure 5.2(a,c,e), designated with the red letters, depict how closely the graph was able to predict the location of the target swarm in steps 30, 50, and 70.

5.1.1 Predictive Value for Graph "East"

The predictive value plot shown in Figure 5.4 shows the predictive value for each step (Section 4.7). In most of the steps before up to about 75, $inclusion_s$, or the proportion of the target swarm that is included by the prediction swarm, is above 0.75. That means that the prediction swarm includes 75% or more of the target swarm in those steps. Also, $discriminatory\_value_s$, or the proportion of predicted occupancy that includes the target swarm, shows that over 75% of the prediction swarm includes the target swarm in those steps. The level of 0.75 is maintained for about 75 steps. By the end of the batch execution we are still able to predict where 50% of the target swarm is and over 50% of the prediction swarm includes the target swarm. The reason for the general drop in $inclusion_s$ after step 75 is that $total\_occupancy_s = \sum O_p^s$ overtakes $total\_predicted_s = \sum \hat{O}_p^s$ in step 70. This can be seen in the predictive value supplement plot in Figure C.1. The prediction swarm continues to outnumber the target swarm for the remainder of the batch execution as the target swarm continues to exit at the edge of the world. That $total\_predicted_s$ has surpassed $total\_occupancy_s$ can also be seen in the excluded/overestimate heat maps in Figure C.3 where we see that the heat map goes from being predominantly blue (more regions where $e_p^s > 0$) in steps 30 and 50 to being predominantly red (more regions where $o_p^s > 0$) in step 70. The predictive value supplement plot in Figure C.1 also shows that $included_s$ (green) remains fairly constant, as does $excluded_s$ (yellow) and $overestimate_s$ (magenta). But while $excluded_s$ and $overestimate_s$ are relatively constant throughout batch execution, $included_s$ decreases with $total\_occupancy_s$, as there is less target swarm to include. With this decrease in $included_s$, the gap between $included_s$ and both $excluded_s$ and $overestimate_s$ decreases and with that gap decrease comes the decrease in $inclusion_s$ and $discriminatory\_value_s$ that can be seen in the predictive value plot in Figure 5.4. The decrease in $inclusion_s$ and $discriminatory\_value_s$ happens because $included_s$ decreases while $excluded_s$ and $overestimate_s$ largely remain the same (Equation 4.10 and Equation 4.11).

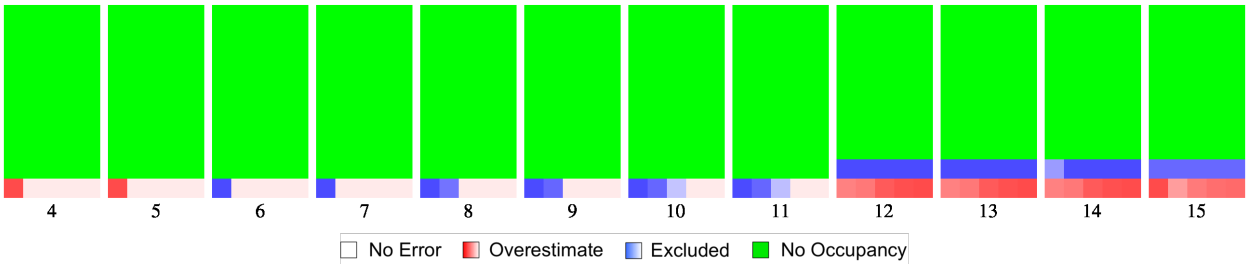

Figure 5.5: *Normalized excluded and overestimate heatmaps for graph "East" in steps 4-15.*

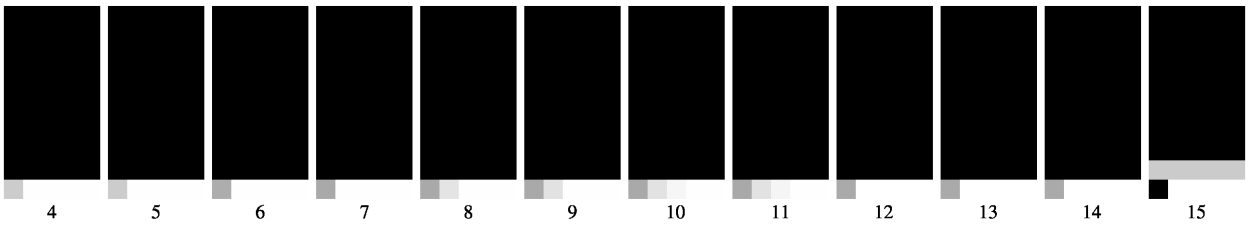

Figure 5.6: *Normalized included heatmaps for graph "East" in steps 4-15.*

While the batch execution ran for 100 steps, 4 steps are missing from the beginning of the predictive value plot (Figure 5.4) because we found that the prediction occupancy matrix for step s predicted the target occupancy matrix for step $s+4$ better than the target occupancy matrix for step s. As can be seen in the predictive value plot in Figure 5.4, $inclusion_s$ and $discriminatory\_value_s$ can oscillate. This is due, in part, to the fact that we discretize the (already discrete in this case) world into a discrete space and discrete time, and the movement of the prediction and target swarms are not always perfectly in phase. By shifting the comparison by four steps, we are able to bring them more into phase. The predictive value in Figure 5.25 for graph "West2 A" (Section 5.4), Figure 5.44 for graph "West3 A" (Section 5.6), Figure 5.53 for graph "West3 B" (Section 5.7), and Figure 5.90 for graph "West1 B" (Section 5.11) are better examples of oscillations in $inclusion_s$ and $discriminatory\_value_s$.

The predictive value plot for graph "East" (Figure 5.4) demonstrates the ability of the predictive value to capture the predictions. For example, in steps 4 and 5, $inclusion_s$ (green)

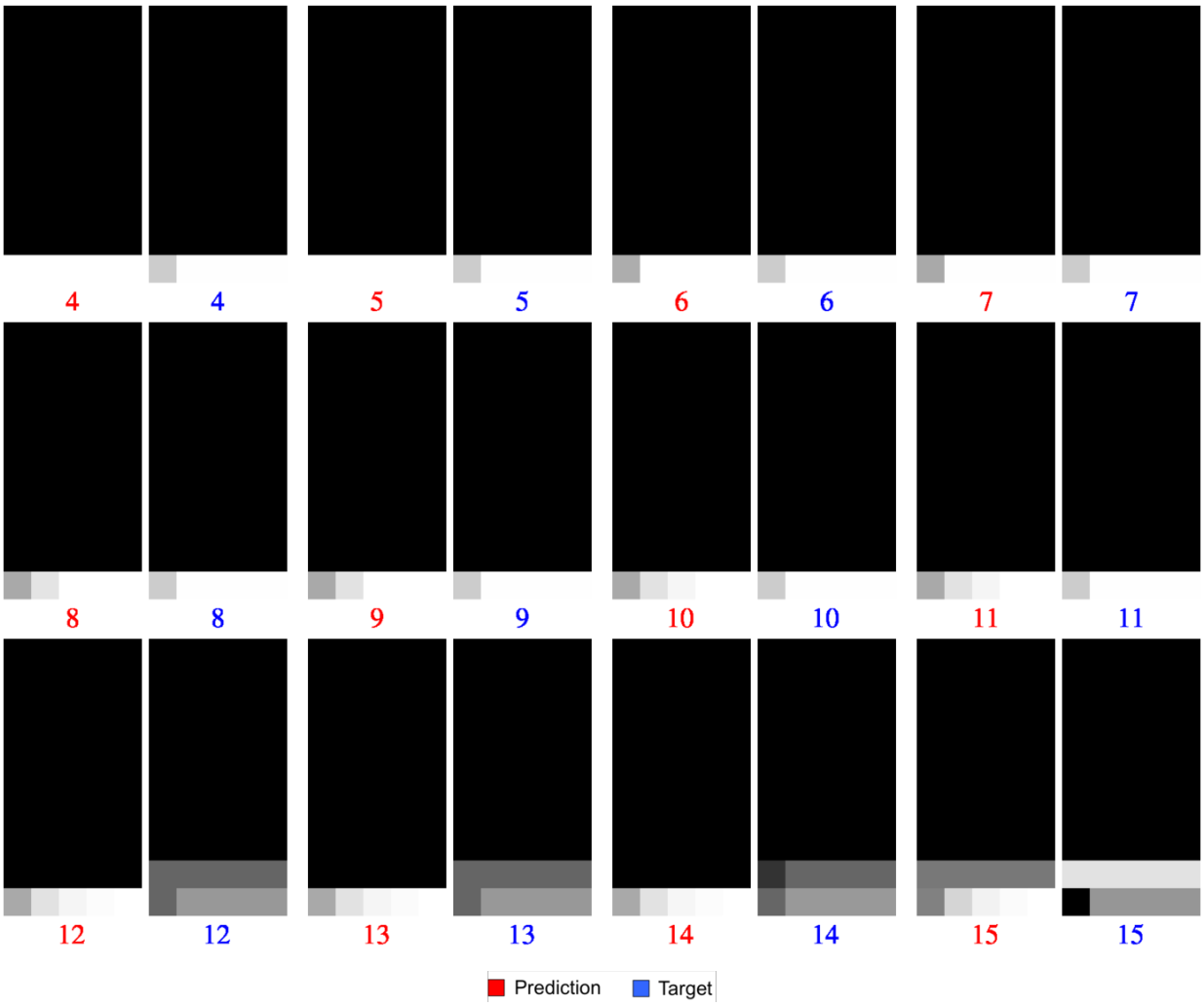

Figure 5.7: *Predicted occupancy for graph "East" in steps 4-15.*

is higher than $discriminatory\_value_s$ (red) and in steps 6 and 7 $discriminatory\_value_s$ is higher than $inclusion_s$, although they are close in all of those steps. Then $inclusion_s$ steadily drops over steps 8-11 before dropping precipitously in step 12. Figure 5.5 shows what is happening. In steps 4 and 5, $o_p^s > 0$ (the prediction overestimates in all regions) and all of the target swarm is included, but not all of the prediction swarm includes the target swarm and so $inclusion_s$ is 1 and $discriminatory\_value_s$ is below 1. In steps 6 through 11 $overestimate_s$ (red) decreases with the increase in $excluded_s$ (blue), which is much higher (it is much darker) than $overestimate_s$. With this increase in $excluded_s$ comes a decrease in $included_s$, decreasing $inclusion_s$ and $discriminatory\_value_s$. Finally, in step 12 a large portion of the target swarm moves into row II leading to a large increase in $excluded_s$ there. Then $discriminatory\_value_s$ ends up decreasing along with $inclusion_s$ in step 12 because $included_s$ decreases and $overestimate_s$ increases in row I. This drop in $included_s$ (green) and increase in $excluded_s$ (yellow) and $overestimate_s$ (megenta) in step 12 can be seen in the predictive value supplement plot in Figure C.1. By Equation 4.13 and Equation 4.14, a decrease in $included_s$ will correspond to an increase in $excluded_s$ and $overestimate_s$, respectively, assuming that $total\_predicted_s$ and $total\_occupancy_s$ do not change. From the predictive value supplement plot in Figure C.1, we see that $total\_occupancy_s$ (blue) and $total\_predicted_s$ (red) do not change between step 11 and step 12. Then, by Equation 4.11 and Equation 4.10, both a decrease in $included_s$ and an increase in $excluded_s$ and $overestimate_s$ will result in a decrease to $discriminatory\_value_s$ and $inclusion_s$. Finally, note that by Equation 4.10 and Equation 4.12, $exclusion_s$ (blue) will always mirror $inclusion_s$ (green) around 0.5, which is what we see in the predictive value plot (Figure 5.4).

The occupancy for steps 4 through 15 in Figure 5.7 explains what is happening in the excluded and overestimate heatmaps for those steps (Figure 5.5). Initially, in step 4, the prediction is spread evenly in row I, but there is actually less of the target swarm in the left-most region (I, I), leading to the greater o_p^s there (red). Then, beginning in step 6, we

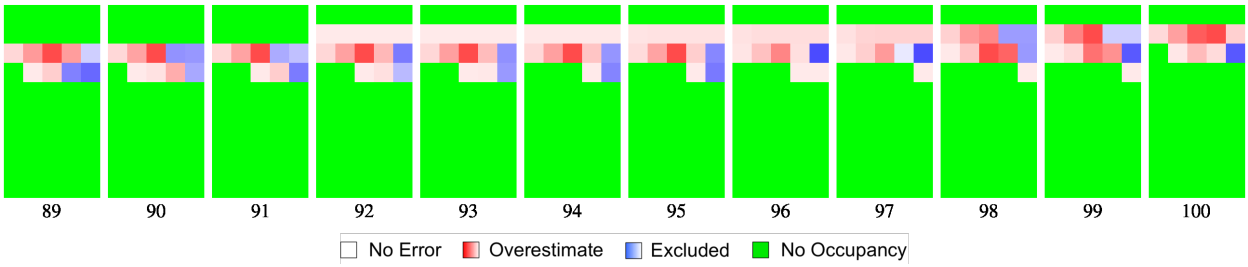

Figure 5.8: *Normalized excluded and overestimate heatmaps for graph "East" in steps 89-100.*

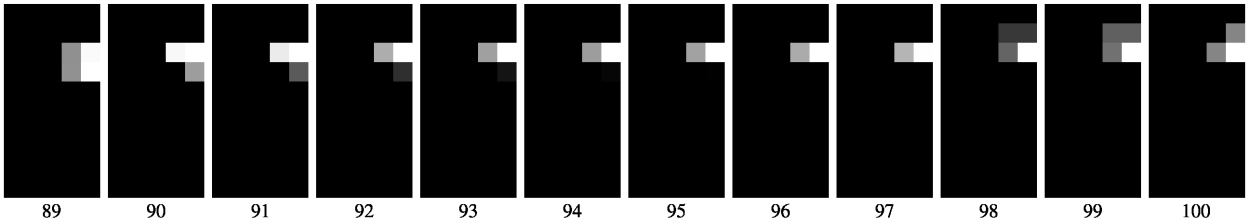

Figure 5.9: *Normalized included heatmaps for graph "East" in steps 89-100.*

see about a third of the prediction swarm moving east every two steps, steadily leaving the regions to the west. This behavior matches the graph in Figure 5.3 where after two steps in the region 32% of the predicted swarm heads east via the virtual virtex "E". The predicted swarm does not move into row II via virtual virtex "N" until the 12^{th} step (step 15 = 12+4-1 = 12-`SHIFT`-1). They first reach "N" in step 15 of the batch execution because the earliest that "N" can be reached is in the 12^{th} step from real node "11", we shift by 4 since step s in the prediction predicts step $s+4$ in the batch execution, and we subtract 1 because the first step is 0. It is this frequent, eastward movement of the prediction that leads $excluded_s$ to increase in the western regions over steps 6 through 11. That $included_s$ is decreasing in the west-most regions in steps 6-11 can be seen in the included heat maps in Figure 5.6. Finally, in step 12, the target swarm moves into row II resulting in the dark blue in row II, while the prediction doesn't move into row II until step 15 resulting in the red in row I. That also leads to the dramatic decrease in $included_s$ (green) and increase in $excluded_s$ (yellow)

and $overestimate_s$ (magenta) in step 12 shown in the predictive value supplement shown in Figure C.1. The included heat maps in Figure 5.6 show how i_p^s goes from looking more like the prediction occupancy in step 11 to looking more like the target occupancy in step 12 (Figure 5.7). This is because $i_p^s = min(O_p^s, \hat{O}_p^s)$ and the target occupancy in the regions goes from being mostly above the prediction occupancy to being mostly below. So i_p^s takes on the characteristics of the lower target swarm occupancy. Thus, the included heat maps agree that $included_s$ decreases in step 12. Of course, there are increases in $included_s$ (green) and decreases in $excluded_s$ (yellow) and $overestimate_s$ (magenta) in steps 15 and 16 when the prediction swarm finally moves into row II from real vertices "11" and "12", respectively. The included heat maps in Figure 5.6 show how i_p^s goes to 0 in (I, I) but becomes positive in row II when the target swarm moves out of (I, I) and the prediction swarm moves into row II.

The excluded/overestimate heatmaps in Figure 5.8 and the occupancy in Figure 5.10 show what is happening in steps 89 to 100 when $inclusion_s$ (green) is higher than $discriminatory\_value_s$ (red), $inclusion_s$ is increasing, and $discriminatory\_value_s$ decreasing in the predictive value plot in Figure 5.4. The plot shows $inclusion_s$ and $discriminatory\_value_s$ increasing in steps 89, 90, and 91. The excluded/overestimate heatmaps (Figure 5.8) also look very similar for those steps. In step 90, the target swarm and prediction swarm both move into $(IV, VIII)$ and $(V, VIII)$, increasing $included_s$ and decreasing $excluded_s$ and $overestimate_s$. In step 91 even more of the prediction moves into $(IV, VIII)$ and $(V, VIII)$, increasing $included_s$. The included heat maps in Figure 5.9 show that i_p^s increases in $(IV, VIII)$ and $(V, VIII)$ in step 90 and again in $(V, VIII)$ in step 91. Next, the predictive value (Figure 5.4) shows that something changes in step 92 when $inclusion_s$ increases and then decreases steadily until step 97. Again, the excluded/overestimate heatmaps look very similar for those steps. In step 92 there is a significant decrease in $total\_occupancy_s$ (in blue) that can be seen in the predictive value supplement plot in

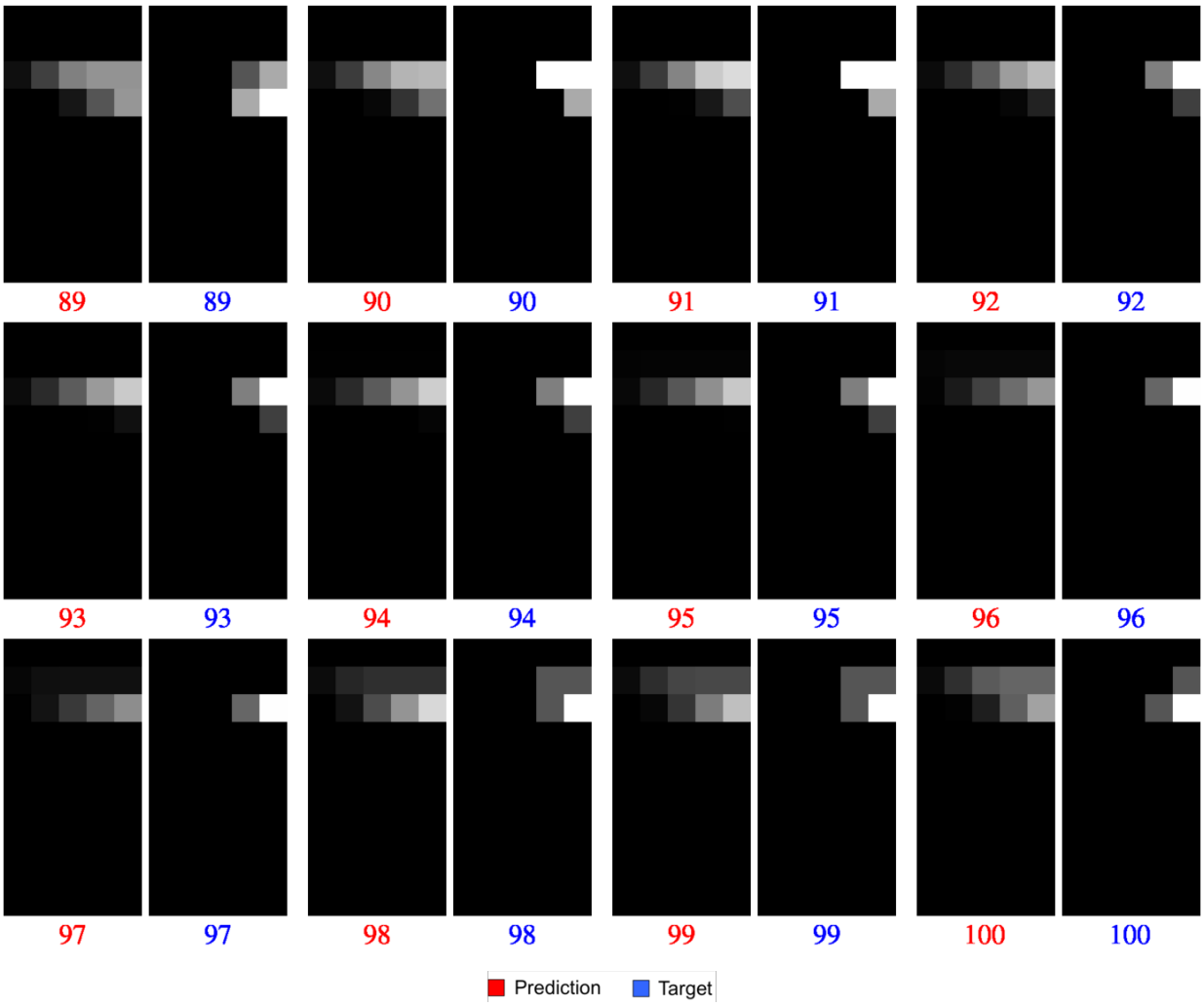

Figure 5.10: *Predicted occupancy for graph "East" in steps 89-100.*

Figure C.1. This drop in $total\_occupancy_s$ decreases $excluded_s$ by a slightly larger amount than in the previous steps thereby increasing $inclusion_s$ by a slightly larger amount than in the previous steps. The drop in $total\_occupancy_s$ and related drop in $included_s$ causes $overestimate_s$ to jump, leading to a drop in $discriminatory\_value_s$ in step 92. Furthermore, in steps 92 through 95 the target concentrates in $(IV, VIII)$ and $(V, VIII)$, decreasing $overestimate_s$ in $(IV, VIII)$ and increasing $excluded_s$ in $(V, VIII)$. This causes the slight $discriminatory\_value_s$ increase from step 92 to step 94. The slight decrease in $discriminatory\_value_s$ in step 95 is due to the slight increase in $overestimate_s$ and even greater decrease in $included_s$ in that step. Then in step 96, $included_s$ decreases and $excluded_s$ and $overestimate_s$ increase when a significant portion of the target swarm moves from (V, VII) into $(V, VIII)$, thereby decreasing $inclusion_s$ and $discriminatory\_value_s$. In step 97, $included_s$ decreases and $overestimate_s$ increases when the prediction swarm moves east and north, leaving additional $excluded_s$ in $(VI, VIII)$ and $(V, VIII)$. The decrease in $included_s$ and increase in $overestimate_s$ decreases $inclusion_s$ and $discriminatory\_value_s$ further. Next, the predictive value in (Figure 5.4) shows an increase and then decrease in $inclusion_s$ (green) and a slight decrease, increase, and then sharp decrease in $discriminatory\_value_s$ (red) in steps 98, 99, and 100. The heat maps in (Figure 5.8) look similar for those steps. In step 98, there is a sharp decrease in $total\_occupancy_s$ (blue) that can be seen in the predictive value supplement plot in Figure C.1. The source of this drop in $total\_occupancy_s$ is $(V, VIII)$ where O_p^s is much higher than $\hat{O}_p^s$, decreasing $total\_occupancy_s$ enough to decrease $excluded_s$ but not enough to decrease $included_s$. That decrease in $excluded_s$ but not $included_s$ contributes to the spike in $inclusion_s$, but not in $discriminatory\_value_s$, in step 98. At the same time both the prediction swarm and target swarm move north into (IV, IX) and (V, IX), contributing further to the increase in $included_s$ and $inclusion_s$, but also $overestimate_s$, resulting in a slight decrease in $discriminatory\_value_s$. In step 99 the prediction swarm shifts east, moving even more prediction into (IV, IX) and (V, IX), in-

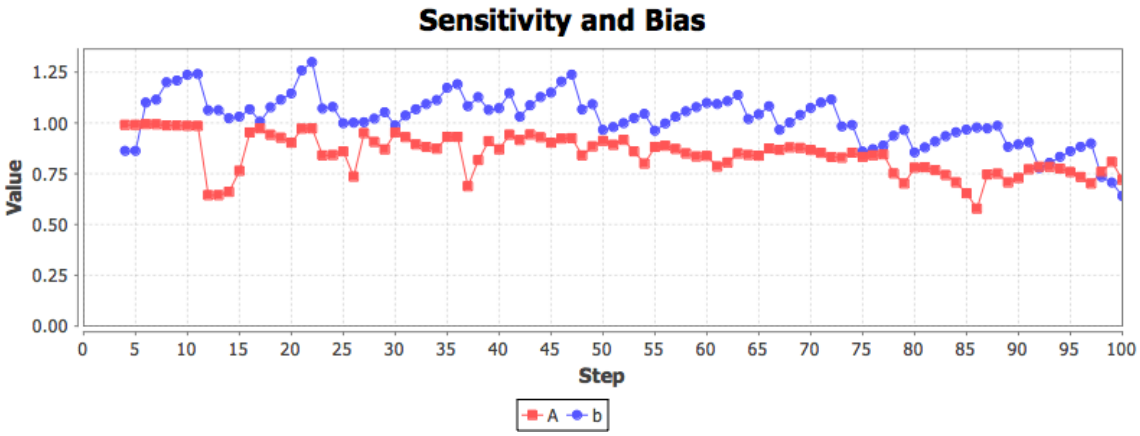

Figure 5.11: *Sensitivity and bias for graph "East".*

creasing both $inclusion_s$ and $discriminatory\_value_s$. Then in step 100, there is a significant decrease in $total\_occupancy_s$ when the target shifts east decreasing $included_s$ and increasing $overestimate_s$ leading to a decrease in both $inclusion_s$ and $discriminatory\_value_s$.

5.1.2 Sensitivity and Bias for Graph "East"

The sensitivity and bias in Figure 5.11 reflect what we see in the predictive value (Figure 5.4) and excluded/overestimate heat maps (Figure 5.5) for steps 4 through 12. There we see we see that $b < 1$ in steps 4 and 5, which corresponds to the relatively high $overestimate_s$ in those steps. Then we see b become greater than 1, signaling a bias toward underestimate, and steadily increase with the growing $excluded_s$ over steps 6 through 11. The predictive value supplement plot in Figure C.1 is in agreement with this. There we see that b changes with the difference between $excluded_s$ and $overestimate_s$. The greater $overestimate_s$ in steps 4 and 5 corresponds to the value of b less than 1. The greater $excluded_s$ in steps 6 through 12 correspond to a value of b greater than 1. Furthermore, the gap between the higher $excluded_s$ and lower $overestimate_s$ grows in steps 6 through 11 corresponding to the increasing b. The gap between the higher $excluded_s$ and lower $overestimate_s$ decreases over

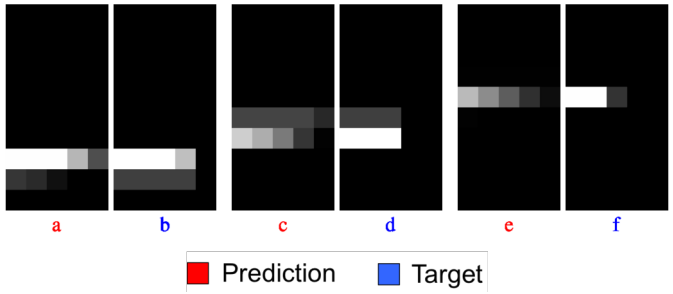

Figure 5.12: *Predicted occupancy for graph "West1 A" at steps 30, 50, and 70.*

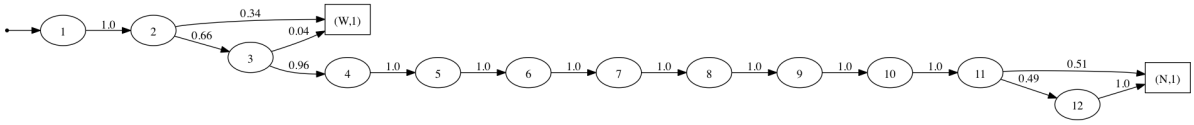

Figure 5.13: *Graph "West1 A" for batch execution "West1".*

steps 11 through 14, corresponding to a decreasing b in those steps. We also see b becoming consistently below 1 on and after step 73 when $overestimate_s$ overtakes $excluded_s$ (first plot in Figure C.1). Furthermore, b continues to represent the distance between $excluded_s$ and $overestimate_s$. The increases in b (getting closer to one and so these are decreases in the overestimate bias) between steps 80 and 88, 89 and 91, and 92 and 97 all correspond to a decrease in that distance between $excluded_s$ and $overestimate_s$ in those steps. Of course, there is a marked increase in the distance between $excluded_s$ and $overestimate_s$ steps 80, 89, and 92 that correspond to the marked drops in b (increases in overestimate bias). In steps 98 through 100 b increases as the gap between $excluded_s$ and $overestimate_s$ grows. The sensitivity is largely in agreement with $inclusion_s$, having a very similar shape. Of course, $inclusion_s$ is the hit rate (Figure 4.10 and Figure 4.18). However, the sensitivity remains above 0.75 even when $inclusion_s$ drops to between 0.5 and 0.75 after step 75, reporting a better performance.

Table 5.4: *Exit analysis for batch execution "West1".*

	Remain			East			North		
Step	Q1	Mean	Q3	Q1	Mean	Q3	Q1	Mean	Q3
1	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.6561	1.0	0.0	0.3439	0.625	0.0	0.0	0.0
3	0.7143	0.9551	1.0	0.0	0.0449	0.0	0.0	0.0	0.0
4	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
9	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
10	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.35	0.4893	0.6	0.0	0.0	0.0	0.4	0.5107	0.65
12	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.9507	1.0

5.2 Predicting a Westward Path

In contrast to the north-east movement in the batch execution "East", the target occupancy matrices for batch execution "West1" in Figure 5.12 show a swarm moving north-west. The world is constructed from square pattern "West1" shown in Figure 4.2(b) and is the vertical reflection of the square pattern "East" (Figure 4.2(a)) used in batch execution "East". We used the statistics in Table 5.4 to produce the graph "West1 A" in Figure 5.13. Since the square pattern for "West1" is the vertical mirror of the square pattern for "East" and the graphs "West1 A" and "East" were created using a similar table of statistics, "West1 A" is very similar to graph "East" (Figure 5.3). The only major difference between "West1 A" and "East" is that the robots exit west in "West" where they exited east in "East" since the virtual vertex "E" has been replaced by the virtual virtex "W".

The predictive value is shown in Figure 5.14. Notice how it closely resembles the predictive value for graph "East" for batch execution "East" in Figure 5.4. This is to be expected since graph "West1 A" is very similar to graph "East" since the batch execution

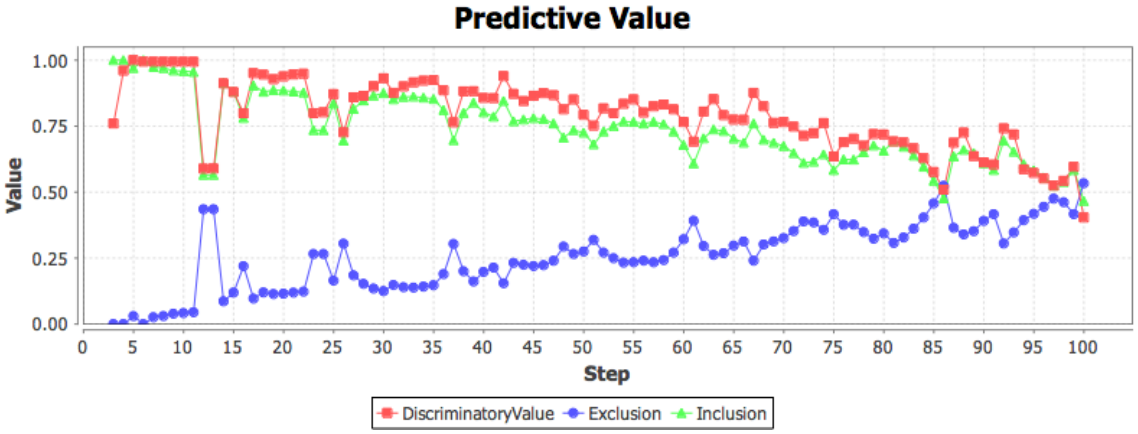

Figure 5.14: *Predictive value for graph "West1 A".*

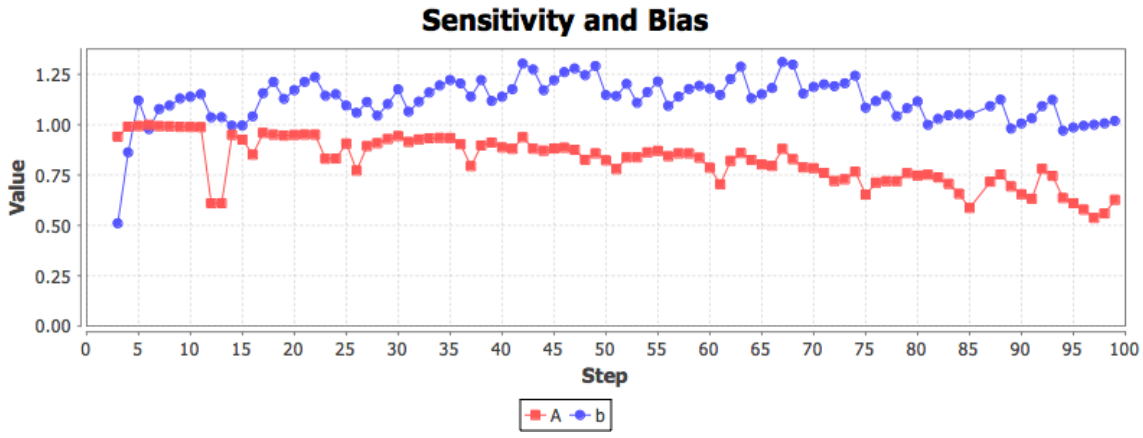

Figure 5.15: *Sensitivity and bias for graph "West1".*

"West1" is nearly a vertical mirror of "East". The low $discriminatory\_value_s$ in steps 3 and 4 is due to very high $overestimate_s$ as suggested by the value of b for those steps in Figure 5.15. This is also apparent in Figure 5.16 where we see only red (overestimate) in the occupied regions. In addition, Figure 5.17 shows that the prediction swarm occupancy is much higher than the target swarm occupancy in all regions in step 3 and at least as high in step 4. The drop in predictive value at steps 12 and 13 is due to the fact that the prediction can be out of step with the target for a few steps, although it may remain in sync for much of the execution. In fact, Figure 5.16 and Figure 5.17 reveal that the prediction is off by two steps. The target swarm moves into row II in step 12, but the prediction swarm doesn't move into row II until step 14. This happens to be the worst such drop in $inclusion_s$ and $discriminatory\_value_s$ for this batch execution. This drop in predictive value occurs because when a large portion of the target swarm moves into row II, e_p^s rises in the regions in row II, raising the $exclusion_s$, and causing a decrease in $inclusion_s$. This is because the target swarm occupancy is higher than the prediction swarm occupancy in row II. In addition, $discriminatory\_value_s$ decreases because the prediction swarm occupancy is higher than the target swarm occupancy in row I, increasing $overestimate_s$. The predictive value is restored when the prediction occupancy matrix $\hat{O}^s$ is once again in sync with the target occupancy matrix O^s in step 14. The fact that the sensitivity and bias are not defined in steps 86 and 100 (see Figure 5.15) can be explained by the fact that the false-alarm rate exceeds the hit rate when the predictive value drops below 50% in those steps since $inclusion_s$ drops below 50% and $inclusion_s$ is the hit rate (see the hit rate in the sensitivity and bias supplement plot in Figure D.4). This happens because, as can be seen for step 100 in Figure 5.18 and Figure 5.19, the target is concentrated in one or two regions and the prediction is spread thinly across these and several others leading to a high $overestimate_s$ and $excluded_s$ and hence a low $inclusion_s$ and $discriminatory\_value_s$. The jump in $discriminatory\_value_s$ and $inclusion_s$ in steps 92 and 93 can be attributed to the

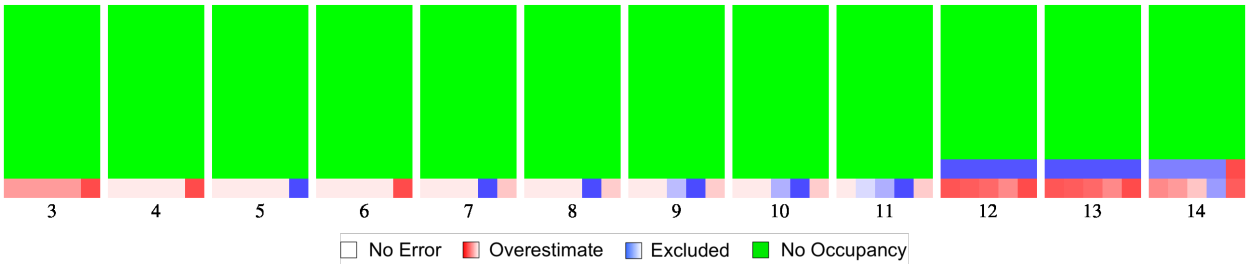

Figure 5.16: *Normalized excluded and overestimate heatmaps for graph "West1 A" in steps 3-14.*

movement of the target and prediction north and west. The target moves out of (I, VII) and (II, VII) and into $(I, VIII)$ and $(II, VIII)$ in step 92 (see Figure 5.19). At the same time the prediction moves out of (I, VII) and into $(I, VIII)$ in step 92. The associated drop in $excluded_s$ and $overestimate_s$ in step 92 (Figure 5.18) results in a significant increase in $inclusion_s$ and $discriminatory\_value_s$. Note that $overestimate_s$ in row IX can be considered insignificant since the values are so low. The prediction is not even visible in row IX in the occupancy (see Figure 5.19). Then, the target shifts west in step 94 reducing i_p^s in $(I, VIII)$ and replacing e_p^s with o_p^s in $(II, VIII)$, reducing i_p^s there as well, resulting in the decrease in $inclusion_s$ and $discriminatory\_value_s$. The reduction in $total\_occupancy_s$ and associated increase in $excluded_s$ and $overestimate_s$ in step 94 can also be seen in Figure C.4 in the predictive value supplement plot. It turns out that the prediction occupancy matrix for step s in batch execution "West1" predicts the target occupancy matrix for step $s + 3$ better than the target occupancy matrix for step s. By shifting accordingly, the oscillations are reduced in the predictive values and the values tend to be higher overall. We revisit shifting in Section 5.10.

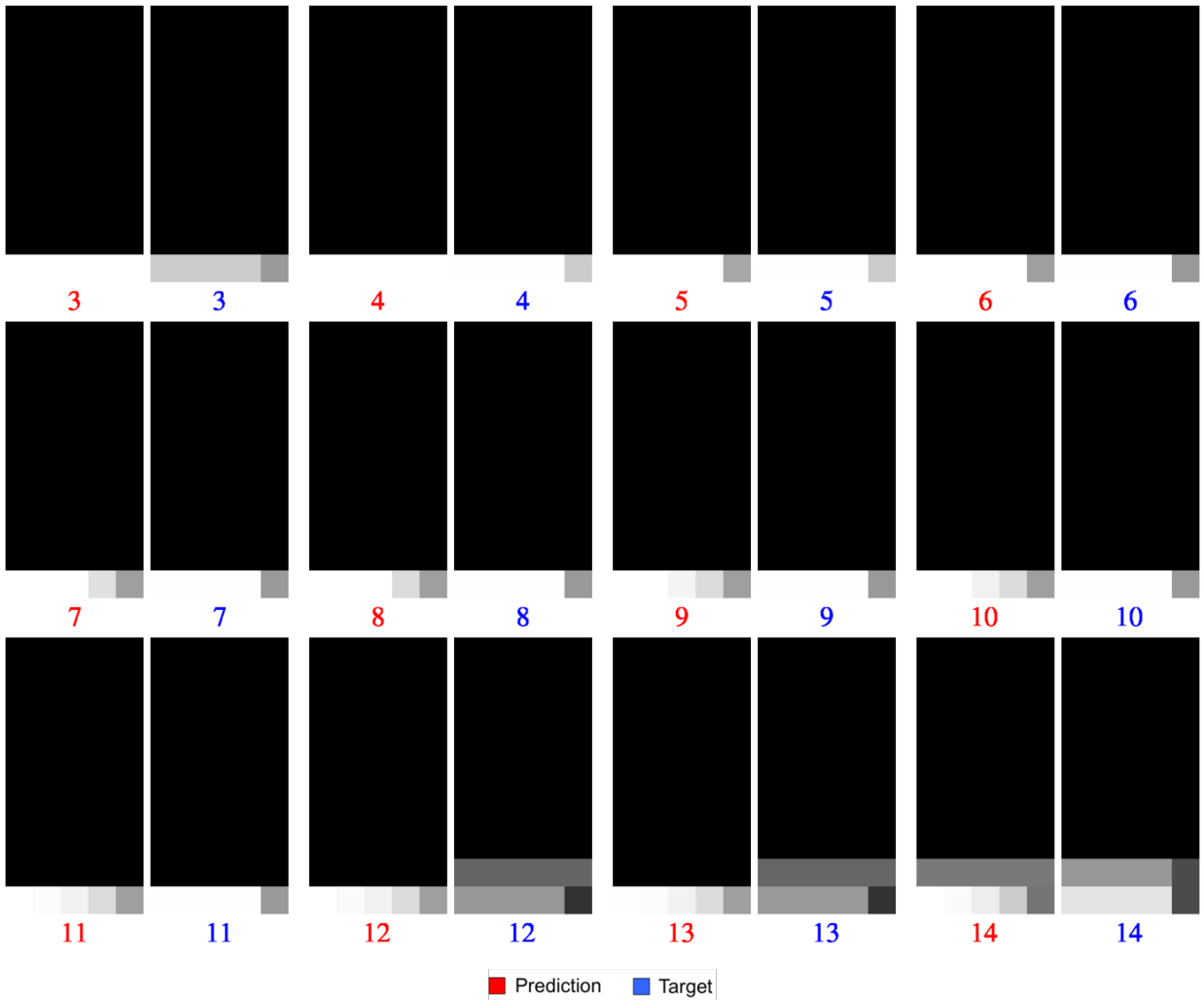

Figure 5.17: *Predicted occupancy for graph "West1 A" in steps 3-14.*

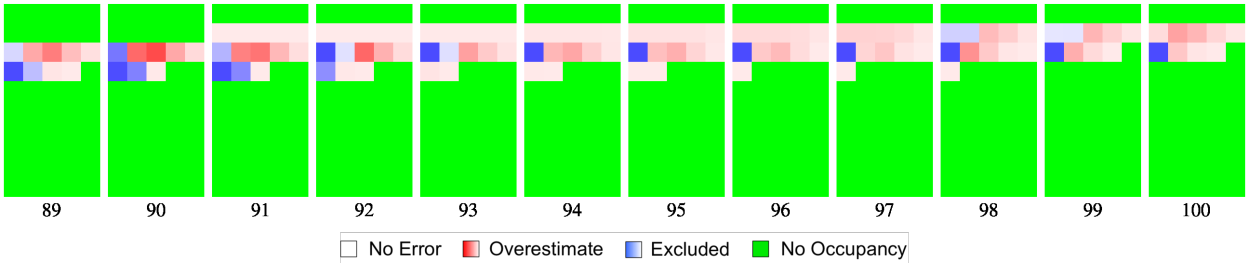

Figure 5.18: *Normalized excluded and overestimate heatmaps for graph "West1 A" in steps 89-100.*

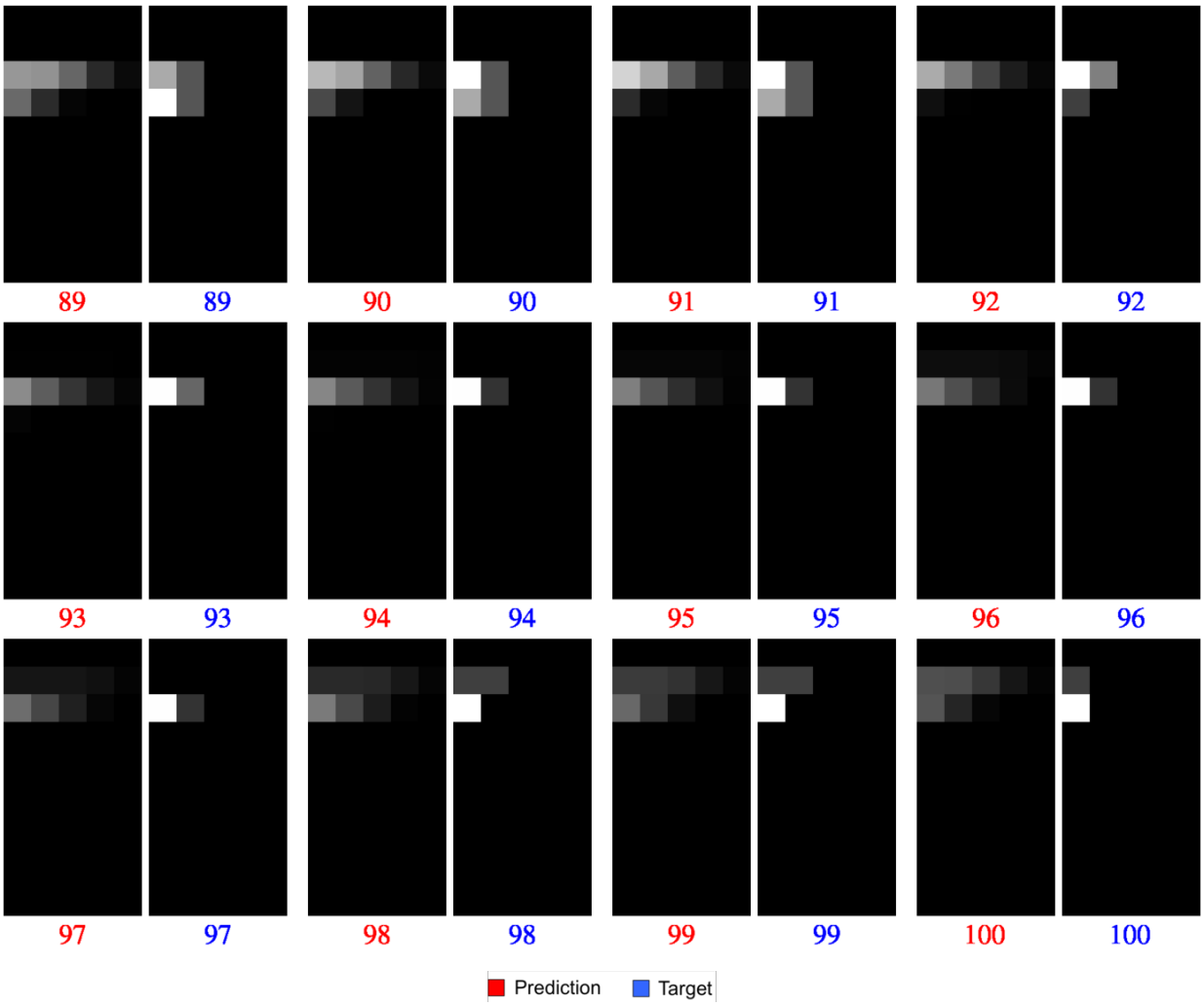

Figure 5.19: *Predicted occupancy for graph "West1 A" in steps 89-100.*

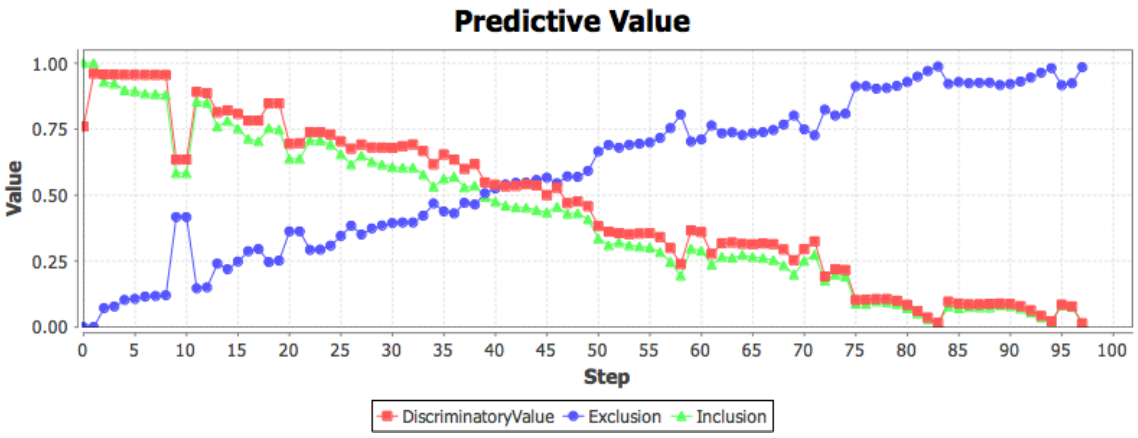

Figure 5.20: *Predictive value of graph "West1 A" for batch execution "East".*

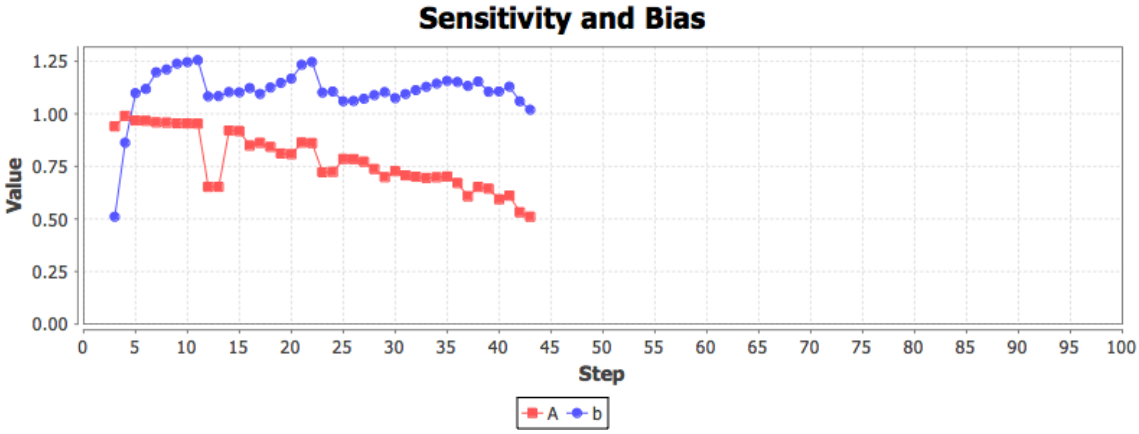

Figure 5.21: *Sensitivity and bias of graph "West1 A" for batch execution "East".*

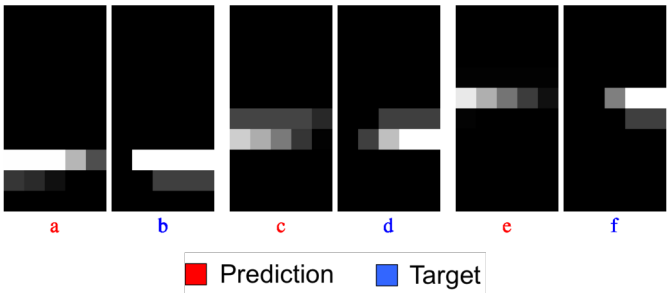

Figure 5.22: *Predicted occupancy of graph "West1 A" at steps 30, 50, and 70 for batch execution "East".*

5.3 Predictive Value Demonstration

While similar matrices produce a high predictive value, diverging matrices result in decreasing predictive value. An attempt to predict occupancy in batch execution "East" with the graph for batch execution "West1" results in the predictive values plotted in Figure 5.20. The inclusion and discriminatory value remain above 0.5 for about half of the of batch execution while the distributions happen to have some overlap. Afterward they drop below 0.5 for the rest of the batch execution as the target and prediction continue down divergent paths. Later, in the last steps, the plot shows that the graph is able to predict the location of well under 25% of the swarm and well under 25% of the predictions include the swarm. The values are, however, above 0.0 as there is still some overlap between the predicted and target occupancy matrices as in column III in step 70. This can be seen in Figure 5.22. The sensitivity and bias are shown in Figure 5.21. They are only defined for the first 44 steps, since the false-alarm rate exceeds the hit rate in step 45. This can be seen in the sensitivity and bias supplement plot in Figure D.6. The related occupancy is in Figure 5.22. The divergence in the target and prediction occupancy matrices corresponds to the divergence in the predictive value. The excluded/overestimate heat maps in Figure C.9 demonstrate the underestimate bias b in step 30 with a large amount of red indicating values of $o_p^s > 0$ in those regions. The excluded/overestimate heat maps also reveal how the prediction swarm went west and the target swarm east since the overestimate (red) is increasingly on the west side and the underestimate (blue) is increasingly on the east side. The underestimate bias is also very visible in the predictive value supplement plot in Figure C.7 where it can be seen that $excluded_s$ is quite a bit higher than $overestimate_s$. This also results in $discriminatory\_value_s$ being higher than $inclusion_s$. Nevertheless, $inclusion_s$ and $discriminatory\_value_s$ remain close because $total\_occupancy_s$ and $total\_prediction_s$ remain close throughout the length of the batch execution (see the predictive value supplement plot

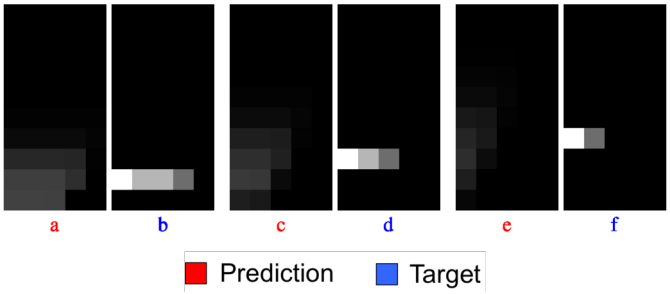

Figure 5.23: *Predicted occupancy of graph "West2 A" at steps 30, 50, and 70.*

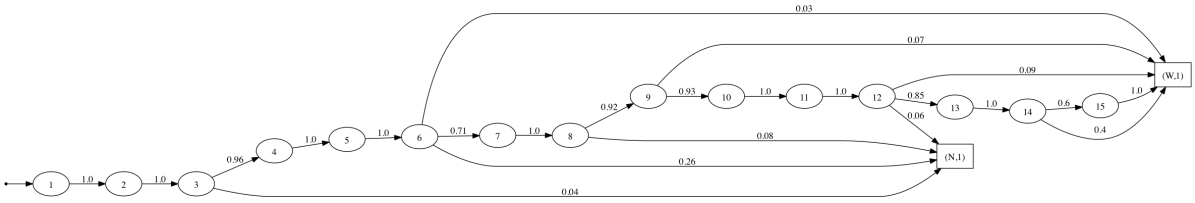

Figure 5.24: *Graph "West2 A" for batch execution "West2".*

in Figure C.7).

5.4 Predicting a Reduced Rate

We now present a prediction for the batch execution "West2" in which the swarm moves left but at a slower rate than for "West1". In fact, the occupancy heat maps for batch execution "West2" in Figure 5.23 show that robots move west but (1) do not make it as far north as they do in batch execution "West1" (see Figure 5.12) and (2) more of the swarm leaves by step 70 than for "West1". The world for batch execution "West2" uses the square pattern in Figure 4.2(c) that is clearly restrictive in that it confines robots to certain regions with impassible lines of targets. It also contains targets, which can be expected to produce dynamics that are more difficult to model given the additional target interference.

In an attempt to construct a prediction for batch execution "West2" we use the statistics in Table 5.5 like we do for "East" and "West1". We use the statistics to produce graph

Table 5.5: *Exit analysis of batch execution "West2".*

	Remain			West			North		
Step	Q1	Mean	Q3	Q1	Mean	Q3	Q1	Mean	Q3
1	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.9495	1.0	0.0	0.0	0.0	0.0	0.0227	0.0833
4	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.7778	0.7074	1.0	0.0	0.0333	0.1	0.0	0.2593	0.5
7	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1.0	0.9234	1.0	0.0	0.0	0.0	0.0	0.0762	0.2857
9	1.0	0.9333	1.0	0.0	0.0667	0.25	0.0	0.0	0.0
10	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
11	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.8	0.8489	1.0	0.0	0.0889	0.3333	0.0	0.0622	0.2
13	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
14	1.0	0.6	1.0	0.0	0.4	1.0	0.0	0.0	0.0
15	0.5	0.5	1.0	0.0	0.5	1.0	0.0	0.0	0.0

"West2 A" shown in Figure 5.24. The proportion moving north at step 3 is 0.04 in the graph when it is 0.0227 in the table because we leave out a proportion of 0.0278 in that step that exits toward the south to simplify the representation. Instead, we add the proportion that moves south to the proportion that moves north. We did build a graph where we did not ignore the proportion that went south but it did no better. Since it did no better we chose to stick with the simpler of the two.

Graph "West2 A" results in the predictive value in Figure 5.25. The values $inclusion_s$ and $discriminatory\_value_s$ are 0 in step 0 since $included_s$ is 0 because the target swarm has not yet entered the world (Figure 5.30 and Figure 5.29). The value $inclusion_s$ is 1 and $discriminatory\_value_s$ just above 0.75 in steps 1 through 3 since the prediction swarm occupancy is higher than the target swarm occupancy in row I in step 1 and includes the entire target swarm but it also overestimates. Figure 5.23 reveals why the predictive value is so low overall. While the prediction centers on the target, it is spread thin and

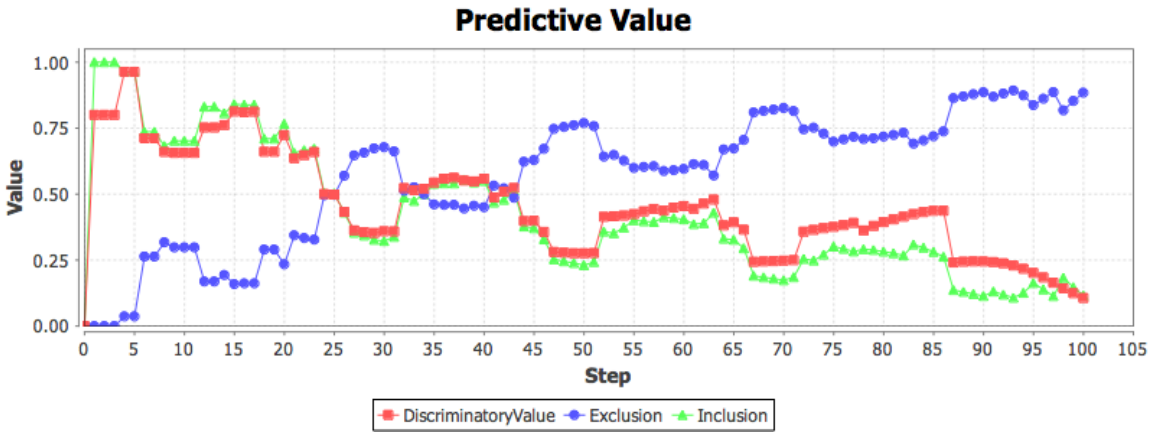

Figure 5.25: *Predictive value of graph "West2 A".*

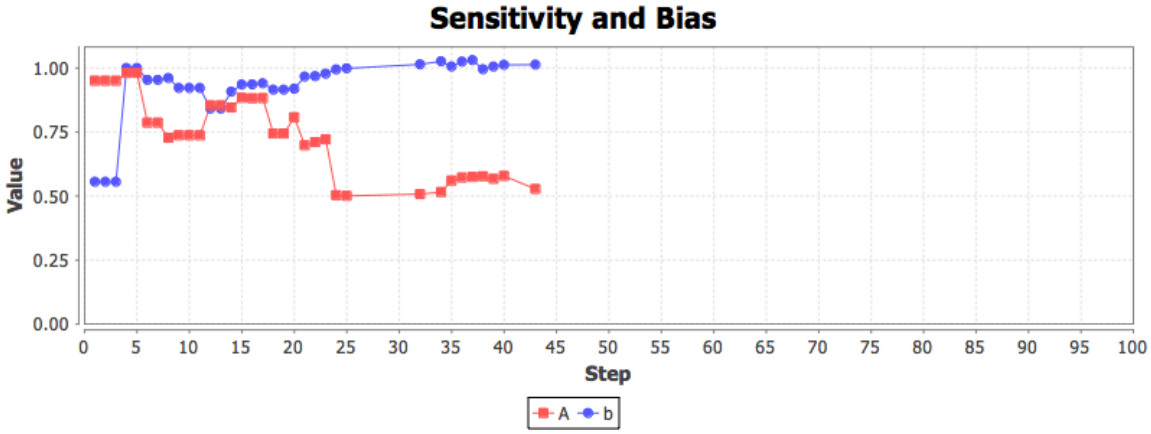

Figure 5.26: *Sensitivity and bias for graph "West2 A".*

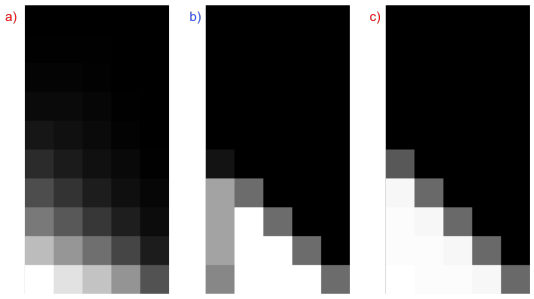

Figure 5.27: *Histogram counts for graphs (a) "West2 A" and (c) "West2 B".*

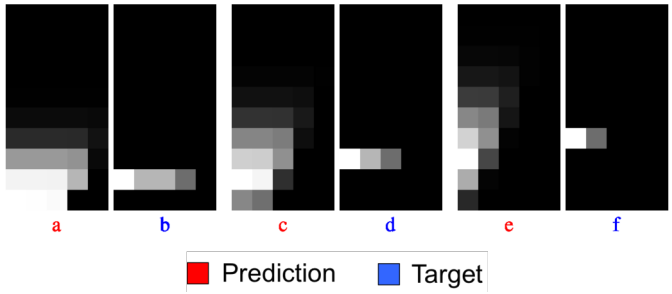

Figure 5.28: *Predicted occupancy of graph "West2 A" at steps 30, 50, and 70. Heat maps were adjusted individually.*

predicts way too low. It misses too much of the target swarm in the most dense regions and overestimates the target swarm occupancy in the ones around them. The adjusted heat maps in Figure 5.28, which make the prediction more visible helping us to compare the matrices, show that the prediction swarm is also about a step behind the target swarm in those steps. Furthermore, the histogram counts for the prediction in Figure 5.27(a) show that the prediction swarm is reaching regions that are unreachable in the world executions. The graph does not properly account for the fact that the targets in the "West2" square pattern create an impassible boundary. Figure 5.31 shows just how spread out the prediction swarm is compared to the target swarm since the red regions (overestimate) cover so much more area and so early while the blue regions (underestimate) cover so little and do not progress nearly as far or nearly as fast. Figure 5.31 and Figure 5.32 explain the oscillations in the predictive value. Figure 5.31 shows the excluded/overestimate heat maps for the alternating local optima in the predictive value. As it happens, the even steps correspond to the local minima and the odd steps the local maxima. Notice how each local minimum corresponds to underestimates that are in fewer regions than for the preceeding local maximum and each local maximum corresponds to underestimates that are in more regions than for the preceeding local minimum. For example, Figure 5.32 shows that in step 37 most of the target swarm is spread over 7 regions for a local maximum. In step 50 most of the target swarm is

spread over only 3 regions for a local minimum, increasing $excluded_s$, decreasing $included_s$, and decreasing the $inclusion_s$ (compared to 7). In step 63 most of the target swarm is spread over 4 regions decreasing $excluded_s$, increasing $included_s$, and increasing $inclusion_s$ (compared to 3). This trend is consistent across the local optima. It is not clear why there appears to be a period of 20 in the predictive value. Without targets, the robots could move through one of the regions within 15 steps. The steps required to proceed through a row may be important since the period is consistent. Hence, it should not depend on something that isn't consistent. Although, in this deterministic world, it is likely that the tagging behavior is repeated in row after row, contributing to the period, and a long period in addition. It is also the case that the targets are one of the biggest differences between this batch execution and the others. In contrast to the predictive value for "West2 A", the periods in the others seem to be more closely tied to the number of steps required to follow the main paths through their regions, further suggesting a role of the targets in the case of "West2 A". Exploration of the periodicity is left for future work. Regions occupied by the target swarm decrease with each successive minimum and with each successive maximum. The prediction swarm continues to spread wider and more thinly. This decreasing number of target regions leads to ever higher $excluded_s$ and $overestimate_s$ and an ever lower $inclusion_s$ and $discriminatory\_value_s$ globally. But while in "West2", due to the hard boundary created by the targets, the target swarm leaves the world quickly and its numbers decrease, decreasing $excluded_s$, the prediction swarm is leaving even faster (see the totals in the predictive value supplement plot in Figure C.10). This leads to even lower value of $overestimate_s$ than $excluded_s$ for steps 52 through 97 causing $discriminatory\_value_s$ to become increasingly higher than $inclusion_s$. Although, toward the end, $total\_occupancy_s$ drops below the prediction total (Figure C.10) and $inclusion_s$ becomes higher than $discriminatory\_value_s$ in steps 98, 99, and 100. Even more, not only is the prediction swarm spread too thin into regions it shouldn't even be in given the target boundaries, but it increasingly fails to center on the target swarm. As

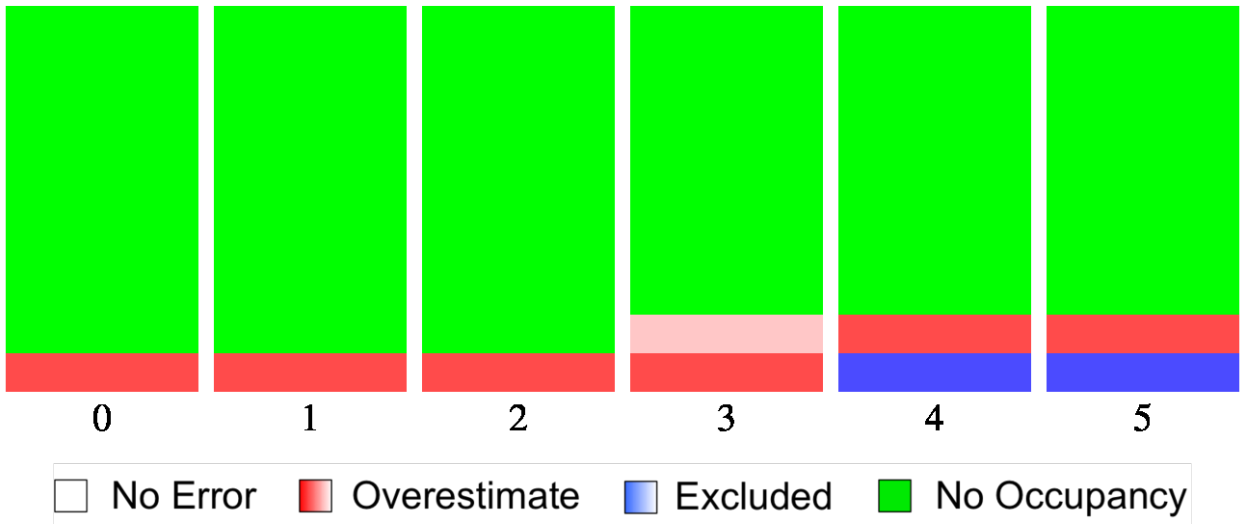

Figure 5.29: *Normalized excluded and overestimate heatmaps for graph "West2 A" for steps 0-5.*

an example, notice how the regions of higher $overestimate_s$ in step 100 are way off the mark and in regions unreachable by the target (see Figure 5.31). The darker shades in the occupancy heat maps in Figure 5.32 reveal how low the values in the prediction occupancy matrix are relative to the target occupancy matrix in step 100. The visibly shaded regions in the prediction occupancy matrix show where the prediction swarm is concentrated and how it is not concentrated where the target swarm is concentrated in step 100.

The sensitivity and bias are shown in Figure 5.26. The sensitivity and bias are undefined in step 0 because the false-alarm rate of 1 is greater than the hit rate of 0 (see the sensitivity and bias supplement plot in Figure D.7) since $included_s$ is 0 (Equation 4.18) and $overestimate_s$ is 1 (Equation 4.19). This is due to the presence of only prediction in the world in step 0. Figure 5.30 shows how there is only the prediction swarm in the world in step 0. Figure 5.29 shows how the fact that there is only the prediction swarm in the world leads to a high overestimate in all of the occupied regions. The sensitivity is just below 1 and the bias just above 0.5 in steps 1 through 3 since some of the target swarm has entered the world but $overestimate_s$ is still higher leading to a reduced sensitivity and an overestimate bias

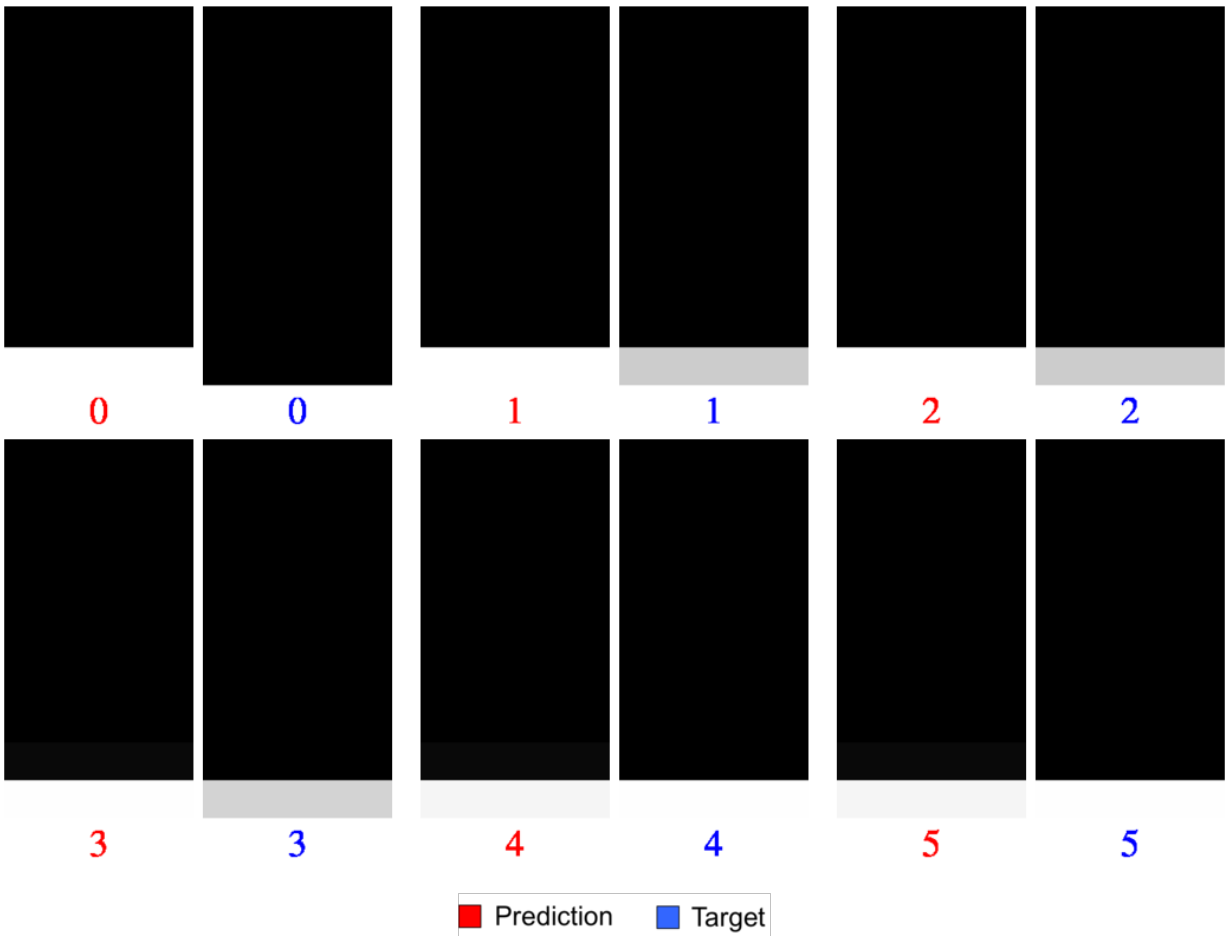

Figure 5.30: *Predicted occupancy for graph "West2 A" for steps 0-5.*

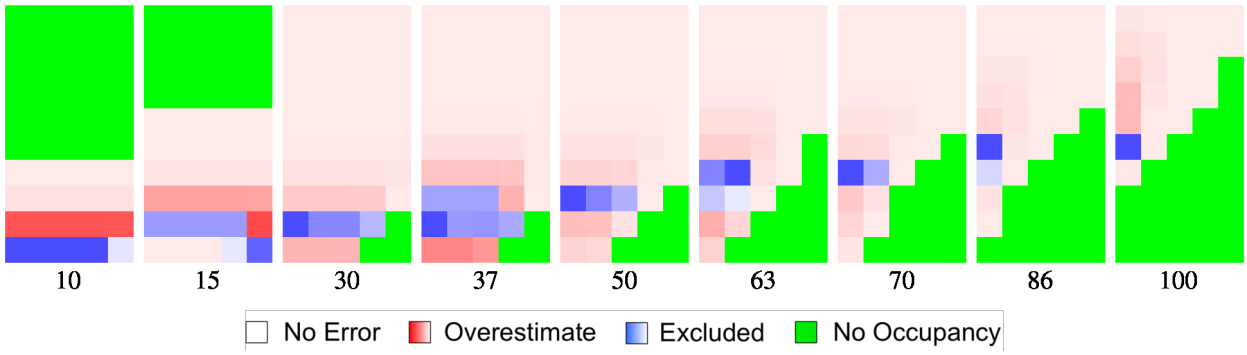

Figure 5.31: *Normalized excluded and overestimate heatmaps for graph "West2 A" for the predictive value optima.*

$b < 1$. The bias b is below 1 before step 30, which corresponds to a higher $overestimate_s$ and lower $excluded_s$ (see the predictive supplement plot in Figure C.10), which in turn corresponds to a higher $inclusion_s$ and lower $discriminatory\_value_s$ (Figure 5.25). The bias is close to 1 between steps 35 and 40 which corresponds to a similar $excluded_s$ and $overestimate_s$, which corresponds to a similar $inclusion_s$ and $discriminatory\_value_s$ in turn.

5.5 Predicting Impassible Boundaries

The poor predictive value associated with the graph "West2 A" (Section 5.4) suggests that we need to account for the impassible boundary created by the "West2" square pattern (Figure 4.2(c)). The way robots move through the square pattern is shown in Figure 5.33. The figure shows how robots move into and out of a region with the square pattern "West2". Robots entering the region from the south cannot exit north and robots entering the region from the east cannot exit west, because they cannot cross the target barrier. By using a graph like "West2 A" (Section 5.24), we are attempting to use a graph that is not divided to model a region that is. By using a disconnected graph we produce a more accurate prediction. The disconnected graph "West2 B" is shown in Figure 5.34. There is still

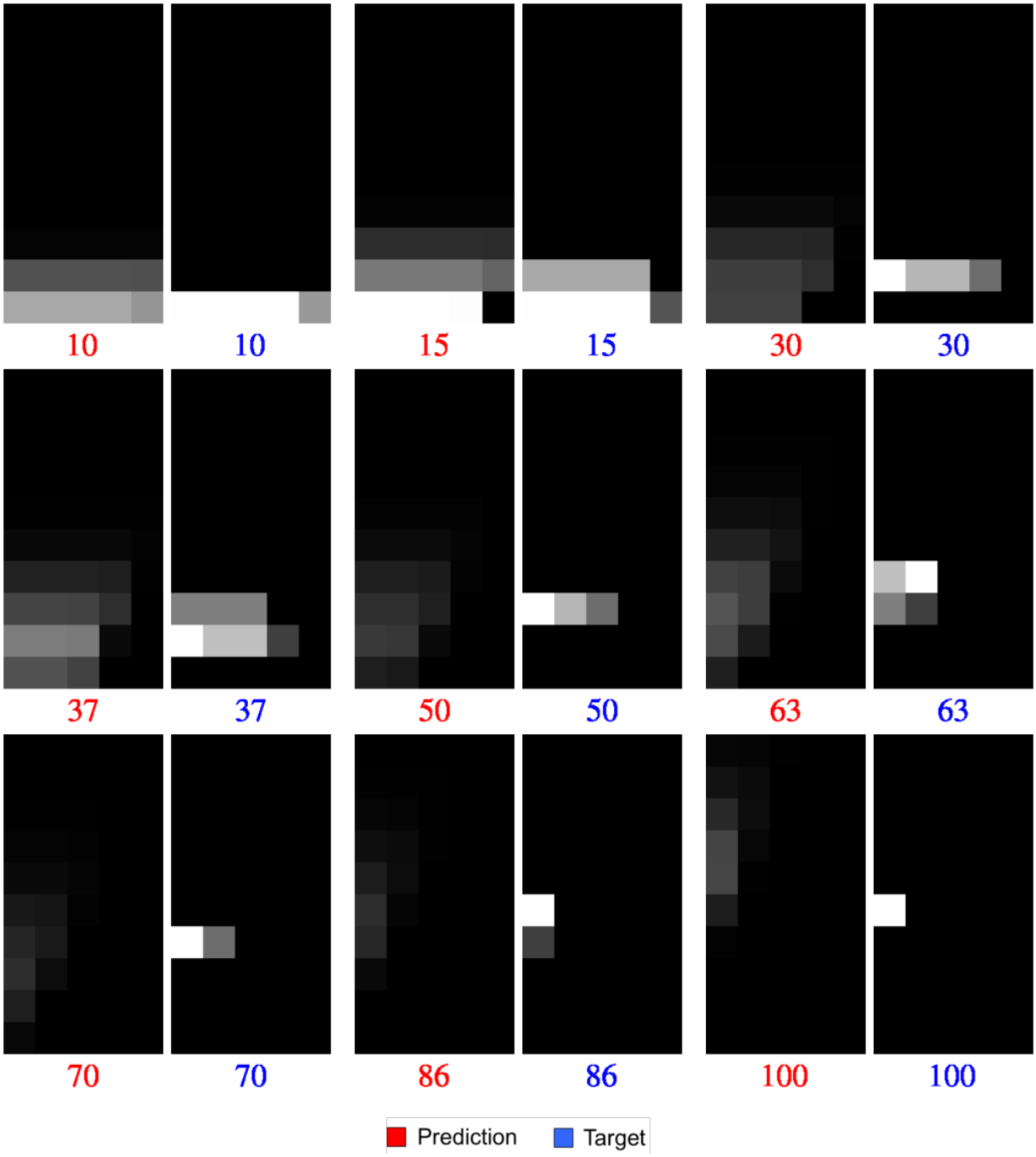

Figure 5.32: *Predicted occupancy for graph "West2 A" for the predictive value optima.*

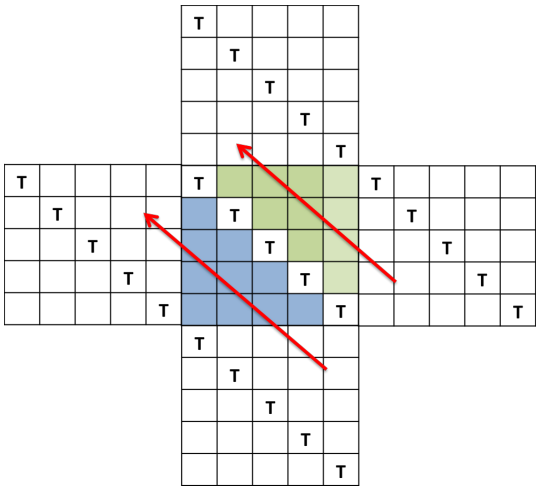

Figure 5.33: *The local behavior for batch execution "West2".*

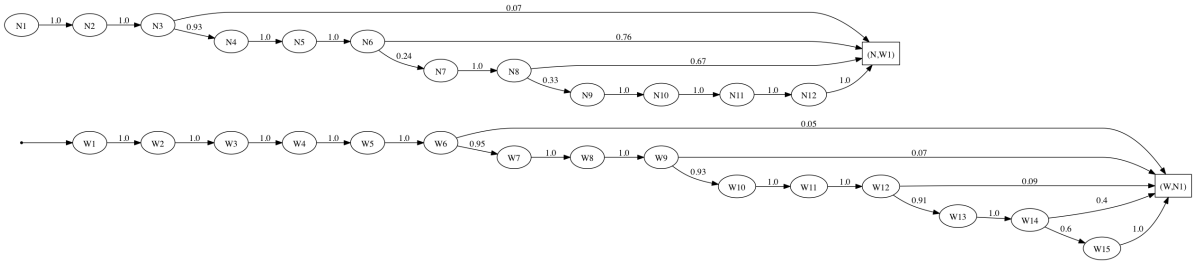

Figure 5.34: *The graph "West2 B" for batch execution "West2".*

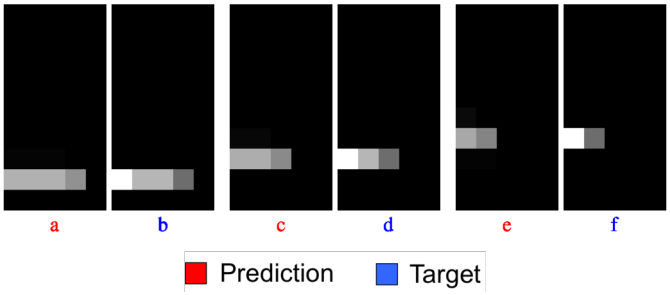

Figure 5.35: *Predicted occupancy for graph "West2 B" at steps 30, 50, and 70.*

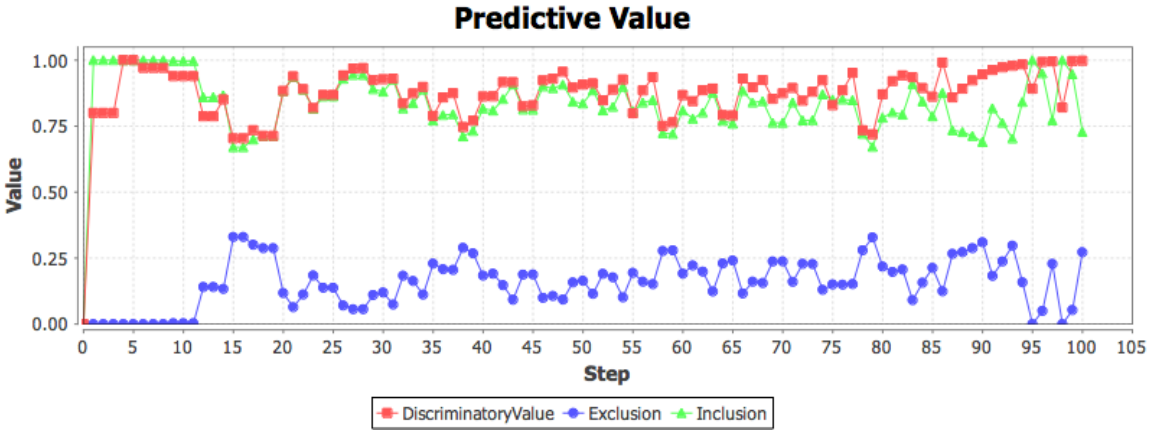

Figure 5.36: *Predictive value for graph "West2 B".*

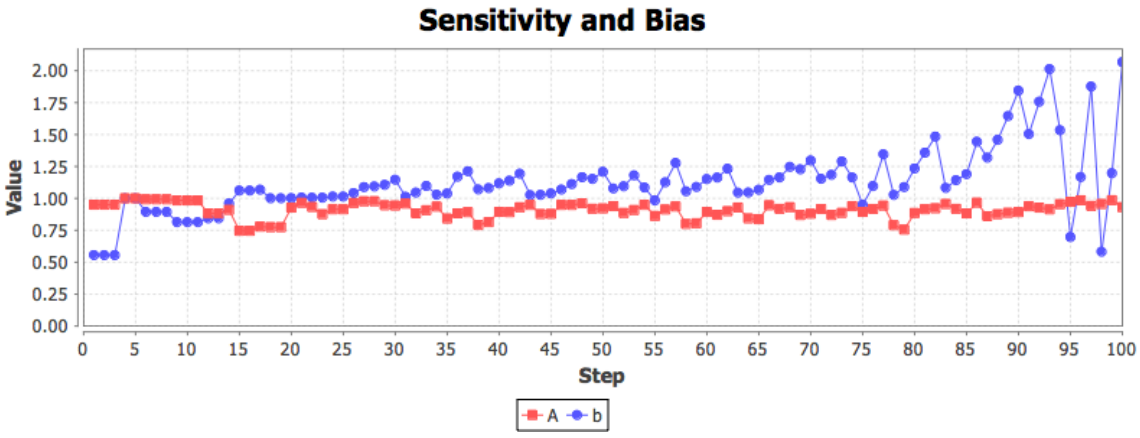

Figure 5.37: *Sensitivity and bias for graph "West2 B".*

only one graph that is used for every region, but the graph is disconnected to reflect the two disconnected zones in the square pattern. The huge improvement can be seen in the occupancy heat maps in Figure 5.35. There we see that the prediction swarm is distributed very much like the target swarm. Figure 5.27(c) shows that the swarm is no longer predicted to cross the impassible barrier. The histogram counts for "West2 B" (c) look very much like those of the batch execution "West2" (b). This means the prediction matches the behavior of the target swarm and the prediction swarm is no longer moving between the disconnected zones in the regions.

The graph in (Figure 5.34) produces a much better predictive value. The step-by-step predictive value is shown in Figure 5.36. The smooth, high $discriminatory\_value_s$ and jagged, lower $inclusion_s$ near the end of the batch execution can be explained by the excluded and overestimate heat maps in Figure 5.38. There we see that, with the exception of steps 95 and 98, most of the prediction includes the swarm. This is because, as can be seen in the occupancy heat maps in Figure 5.39, essentially all of the remaining prediction swarm and target swarm is in (I, V) in all of the steps and since $total\_occupancy_s$ is higher than $total\_predicted_s$ for all but steps 95 and 98 (see the predictive value supplement plot in Figure C.13), there is a very low $overestimate_s$ and very high $discriminatory\_value_s$ in all steps but steps 95 and 98. As for $inclusion_s$, the fact that $total\_occupancy_s$ is higher than $total\_predicted_s$ in all steps but steps 95 and 98 means that there is some amount of $excluded_s$, which reduces $inclusion_s$ in all steps but steps 95 and 98. The amount of $excluded_s$ increases as the $total\_predicted_s$ steadily decreases as the prediction swarm leaves the world but drops sharply in steps 95 and 98 when there is a large decrease in $total\_occupancy_s$. This leads to a steadily rising $inclusion_s$ in all steps but 95 and 98 where there are sharp decreases in $inclusion_s$.

The sensitivity and bias are shown in Figure 5.37. Note that the sensitivity agrees with $inclusion_s$, it only looks less jagged because the values in the plot range from 0 to 2 due to

the very high values of b toward the end that increase the range in the plot. Those jumps in b are explained by the excluded and overestimate heat maps in Figure 5.38. There we see that very few regions are occupied by either the target swarm or the prediction swarm. This fact can be seen in the regional occupancy count plot in Figure C.13 where we see that the total number of occupies regions (red) is 3 until step 96 when it drops to 2. However, it is only three because a small amount of prediction occupied (I, IV) and (II, IV) (cyan) until step 96 (see the excluded and overestimate heat maps in Figure 5.38). But the contribution of the prediction swarm in those two regions in row IV to the sensitivity and bias is negligable. The regional count plot also shows that the total number of regions shared by both the target swarm and prediction swarm (blue) is 1 along with the total number of regions occupied by the target (green). Since the prediction and target swarm effectively occupy and share only (I, V) in steps 89 to 100, changes in the prediction occupancy or target occupancy in (I, V) in steps 89 to 100 have a dramatic effect on b. In step 89, b is well above 1 because e_p^s in (I, V) dwarfs o_p^s in (I, IV) and (II, IV) so that $excluded_s$ is much higher than $overestimate_s$. In step 90 b increases slightly because some of the prediction swarm leaves the world, causing a slight decrease $total\_predicted_s$, which can be seen in the predictive value supplement plot in Figure C.13, and this decrease in $total\_predicted_s$ leads to an increase in $excluded_s$. In step 91, b decreases because the large decrease in $total\_occupancy_s$ causes a large decrease in $excluded_s$ (see the predictive value plot in Figure C.13). The bias b increases steadily in steps 92 and 93 with the decrease in $total\_predicted_s$ and corresponding increase in $excluded_s$. Then in step 94 and 95 there are large decreases in $total\_occupancy_s$ again (Figure 5.37). So much so that with the second decrease in step 95 the prediction swarm outnumbers the target swarm in (I, V), $e_p^s = 0$ and $o_p^s > 0$ and $overestimate_s$ is higher than $excluded_s$, and b dips into the negatives. The bias b increases again in steps 96 and 97 with the decreasing $total\_predicted_s$, since $excluded_s$ becomes higher than $overestimate_s$ in step 96 and increases in step 97. Then in step 98 $total\_occupancy_s$ dips again, once again leaving more prediction

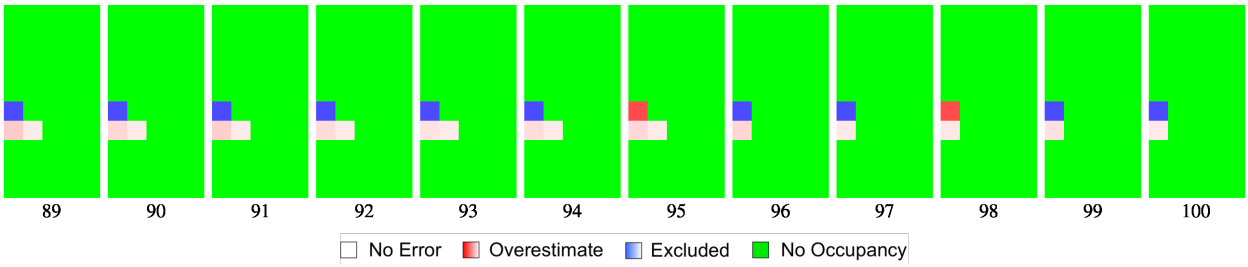

Figure 5.38: *Normalized excluded and overestimate heatmaps for graph "West2 B" for steps 89-100.*

swarm than target swarm in (I, V) and resulting in a negative b. The last two steps see b increasing as more of the prediction swarm leave the world, increasing $excluded_s$.

5.6 Predicting Northward Movement

In another example of predicting divergent behavior, we use the parameter values to produce divergent behavior between two batch executions with the same square pattern. Figure 5.40 shows the occupancy for batch execution "West3" where the swarm moves directly northward through a world constructed from the same square pattern as the world in batch execution "West1" (Figure 4.2(b)). This is very different behavior from what we see in batch execution "West1" where the swarm moved west (Section 5.2). A comparison between the parameter values for "West3" and "West1" in Table 5.2 (numbers 1 and 2) shows that the only difference between the two is that `rtp` is 0.0 for "West1" and 1.0 for "West3".

To see how `rtp` equaling 1.0 can have such a dramatic effect, refer to Figures 5.41 and 5.42 that show how robots move into and out of regions in batch executions "West1" and "West3", respectively. Notice the squares outlined in orange. These are the squares where `rtp` is used to make a decision. In "West1", robots turn left with probability $1 - rtp = 1 - 0 = 1$, and in "West3", robots turn right with probability $rtp = 1$. The end result is the flow depicted in these figures. The purple and green boxes show where robots use `lpp` and `fp`, respectively.

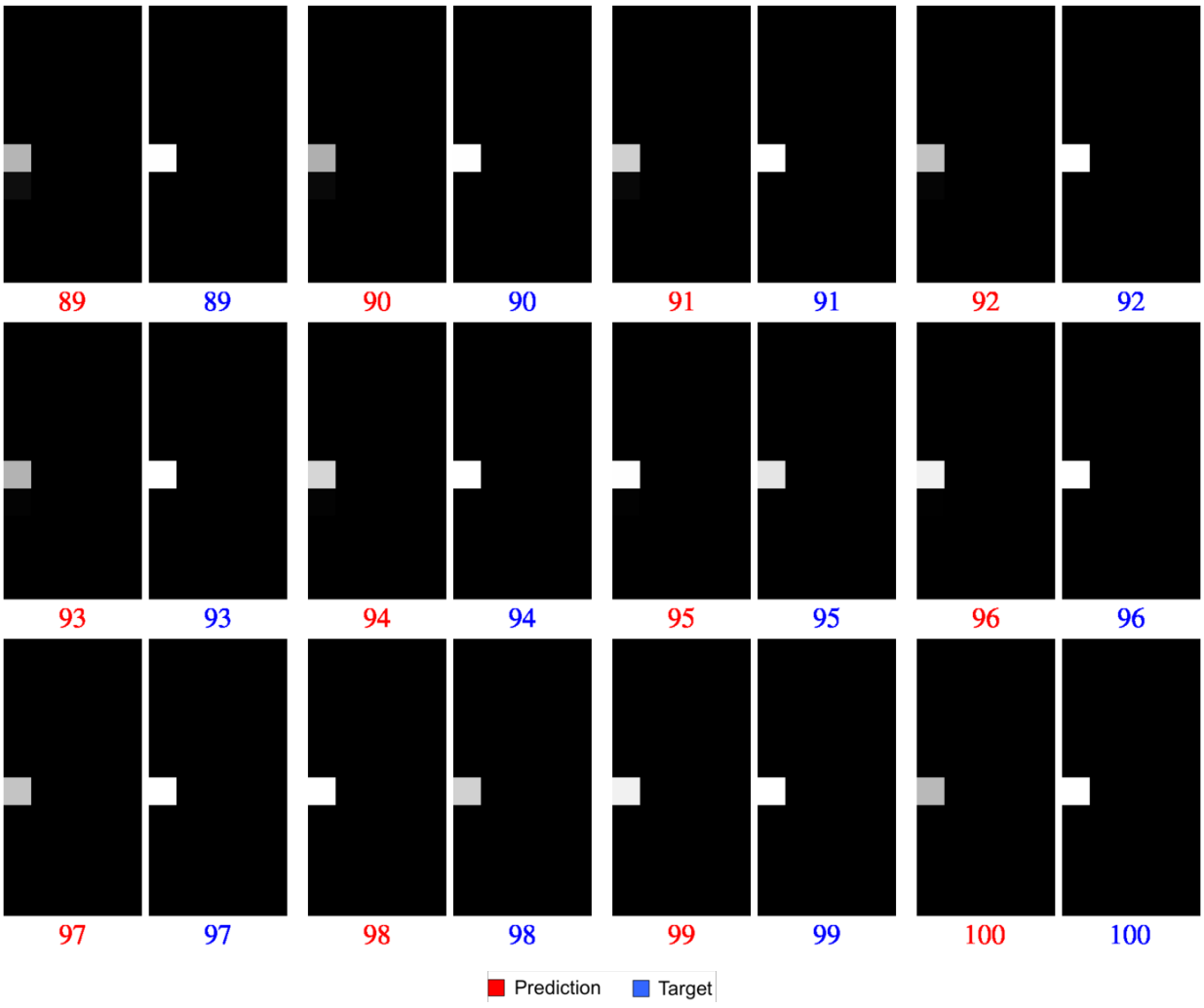

Figure 5.39: *Predicted occupancy for graph "West2 B" for steps 89-100.*

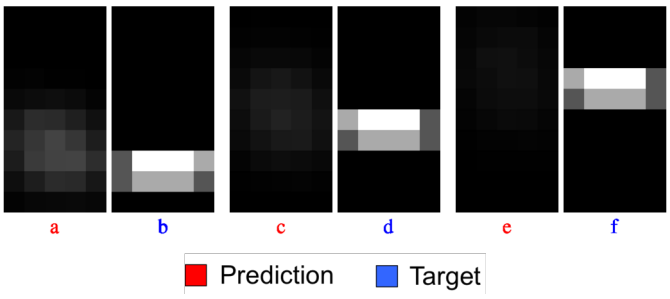

Figure 5.40: *Predicted occupancy of graph "West3 A" at steps 30, 50, and 70.*

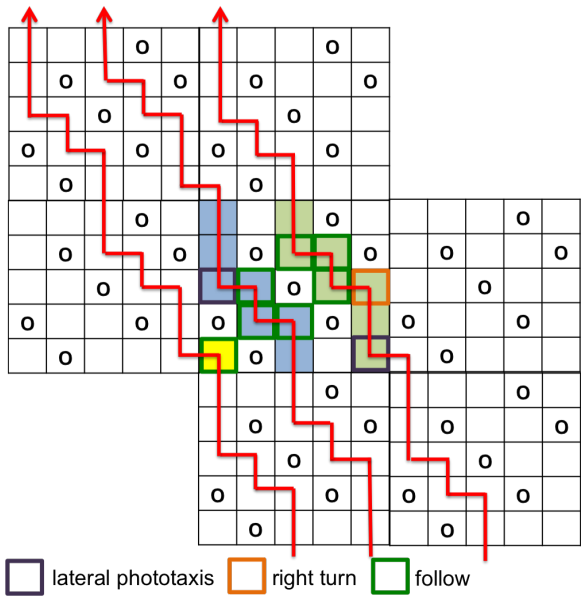

Figure 5.41: *The local behavior in batch execution "West1".*

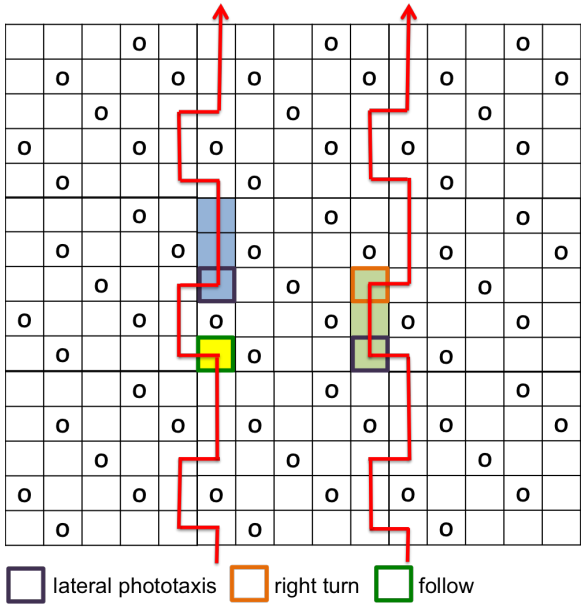

Figure 5.42: *The local behavior in batch execution "West3".*

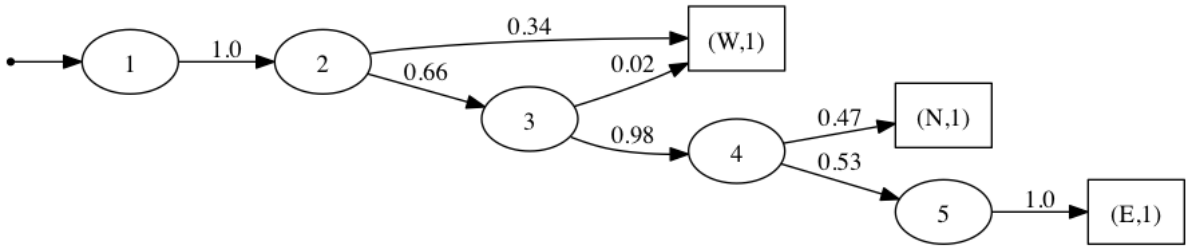

Figure 5.43: *Graph "West3 A" for batch execution "West3".*

Table 5.6: *Exit analysis of batch execution "West3".*

	Remain			West			North			East		
Step	Q1	Mean	Q3	Q1	Mean	Q3	Q1	Mean	Q3	Q1	Mean	Q3
1	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.5278	0.6633	1.0	0.0	0.3367	0.4722	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.9811	1.0	0.0	0.0189	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.5	0.5298	0.625	0.0	0.0	0.0	0.375	0.4702	0.5	0.0	0.0	0.0
5	0.0	0.0703	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.9297	1.0

They do not contribute to the difference between the behaviors in the two batch executions since those probabilities are the same for both "West1" and "West3".

A first attempt at a probabilistic graph for "West3" is the graph "West3 A" shown in Figure 5.43. We use the statistics in Table 5.6. This is a small graph where all robots exit by step 5 rather than by step 12 or 15. However, the predictive value shown in Figure 5.44 is poor. While about 50% of the prediction swarm includes the target swarm in most steps, the prediction swarm includes less than 25% of the target swarm for quite a few steps. The prediction is spread so thin that $inclusion_s$ drops below 25%. That thin spread explains the dark shades in the prediction occupancy heat maps. The heat maps are adjusted individually in Figure 5.46 making the prediction swarm more visible. Note that, while not indicating the exact distribution of the target swarm, the prediction swarm is almost centered on (the prediction swarm seems to be about a row ahead judging from regions with the brightest shades in the prediction heat map) and surrounding the target swarm. The

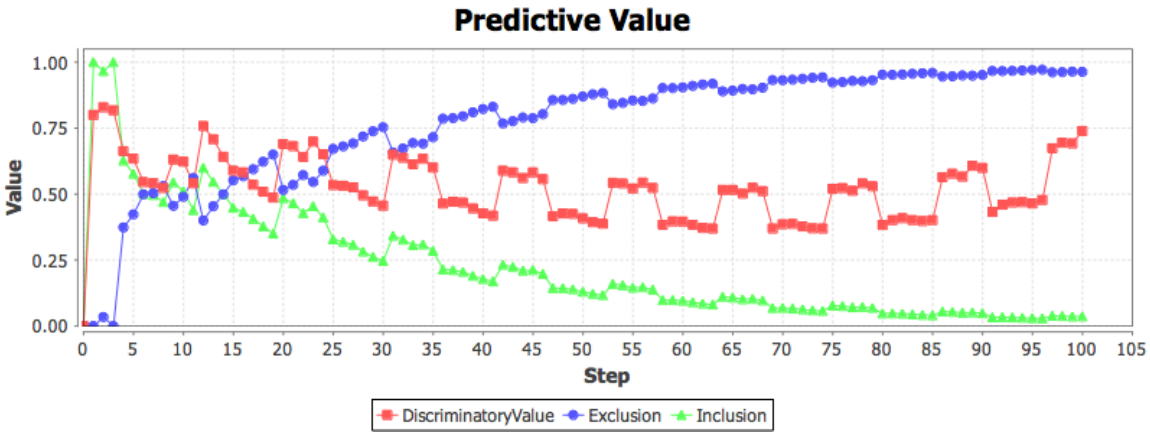

Figure 5.44: *Predictive value for graph "West3 A".*

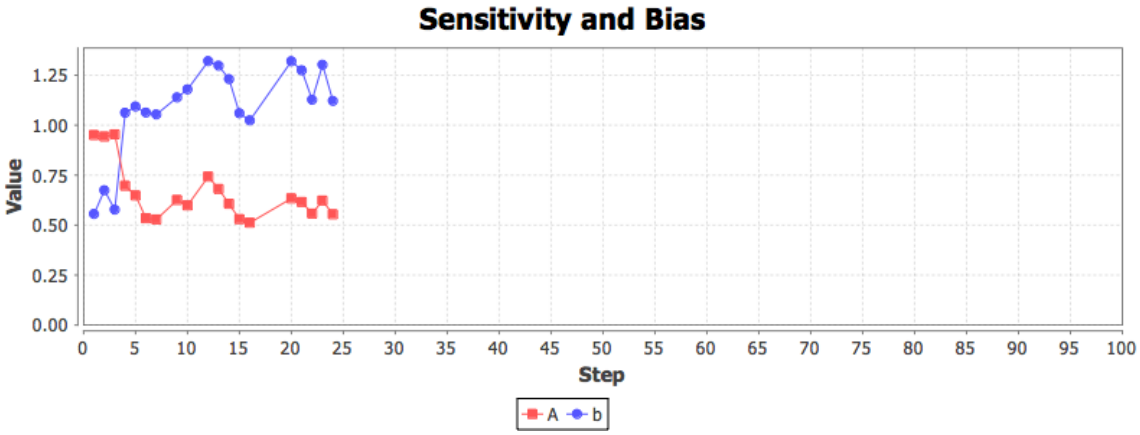

Figure 5.45: *Sensitivity and bias for graph "West3 A".*

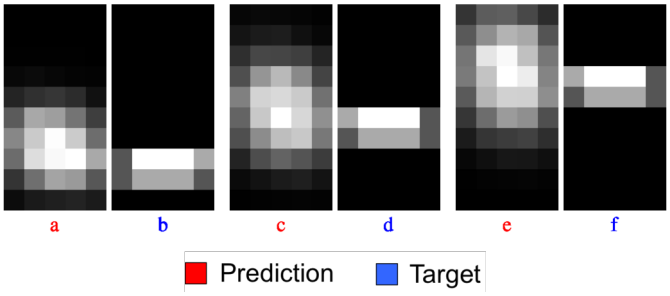

Figure 5.46: *Predicted occupancy of graph "West3 A" at steps 30, 50, and 70. Heat maps were adjusted individually.*

prediction swarm indicates more or less the general location of the target swarm at each of the steps. As for the oscillation in the predictive value, the target swarm advances by moving into a higher row and by moving out of a lower row. However, it moves out of the lower rows a few steps before moving into the higher rows and the target swarm is spread over two rows in some steps and in three at others. Figure 5.50 and Figure 5.51 show selected excluded/overestimate and occupancy heat maps for the local optima. They show both that the prediction is very spread out (there is a large number of red regions) causing $inclusion_s$ to be very small and that the oscillations are due to the target swarm being spread over two rows in some steps and three rows in others. The period is also consistently 11, something that might deserve more attention. The period of length 11 is associated with the fact that the two paths through the rows in the batch execution are 11 steps long. This can be seen in the flow diagram in Figure 5.42. The figure demonstrates the behavior of the target swarm in the world given the square pattern and the probabilities from the batch execution parameters. In addition, the world is deterministic so there is no variation in behavior. There are also no collisions in row II through X in this world. Hence, it takes the target swarm 11 steps to move into or out of a row and when that happens $inclusion_s$ increases or decreases accordingly, since the contribution of the thinly spread prediction swarm to $inclusion_s$ is negligable. We leave an in depth investigation into the periodicity to future work. That the prediction spreads out early can be seen in the excluded and overestimate and occupancy heat maps in Figure 5.48 and Figure 5.49, which reveal just how quickly the prediction swarm spreads (red) out leading to a steep drop in overall $inclusion_s$ and $discriminatory\_value_s$ early on in steps 4 through 8 (Figure 5.44). The excluded and overestimate heat maps (Figure 5.50) show how the regions with overestimate increase and decrease as the target swarm oscillates between 2 and 3 regions (see the occupancy heat maps in Figure 5.51). There is a bit more $included_s$ and quite a bit less $overestimate_s$ when the target swarm is spread over three rows, leading to an increase in $inclusion_s$ and $discriminatory\_value_s$. There is a bit

less $included_s$ and quite a bit more $overestimate_s$ when the target swarm is spread over two rows, leading to a decrease in $inclusion_s$ and $discriminatory\_value_s$. Just how $included_s$ and $overestimate_s$ change with the number of regions occupied by the target swarm can be seen by comparing $included_s$ (green) and $overestimate_s$ (magenta) in the predictive value supplement plot with the shared occupancy (blue) in the regional occupancy count plot in Figure C.16. There it can be seen that $included_s$ increases and $overestimate_s$ decreases when the number of shared regions increases. Oscillations aside, the predictive value supplement plot in Figure C.16 shows why $inclusion_s$ nears 0 while $discriminatory\_value_s$ hovers around 0.5, even increasing toward the end. The figure shows that $total\_occupancy_s$ (blue) does not decrease after step 17, but there continues to be a decrease in $total\_predicted_s$ (red). With the decrease in $total\_predicted_s$ comes an increase in $excluded_s$ (yellow) a corresponding decrease in $included_s$ (green), as there is increasingly less prediction swarm to include the target swarm. This simultaneous large increase in $excluded_s$ and decrease in $included_s$ causes $inclusion_s$ to become very small (Equation 4.10). But $included_s$ and $overestimate_s$ (magenta) decrease together, and a very small $included_s$ divided by the sum of an equally small $included_s$ and $overestimate_s$ does not produce a very small number (Equation 4.11). As $overestimate_s$ continues to alternate with $included_s$ and grow closer to it, $discriminatory\_value_s$ oscillates around 0.5 and even begins to increase. Furthermore, the oscillations in $inclusion_s$ become less pronounced as $included_s$ becomes smaller and smaller relative to such a large $exclusion_s$ (Equation 4.10) but definitely have a much more noticable impact on the $discriminatory\_value_s$ with the much closer $overestimate_s$ (Equation 4.11).

The sensitivity and bias are shown in Figure 5.45. There we see that A is above 0.5 in steps 20 through 24 even though $inclusion_s$ is below 0.5 in those steps (Figure 5.44). The sensitivity and bias supplement plot in Figure D.11 shows that the hit rate is higher than the false-alarm rate in these steps. This is due to the fact that, as the predictive value

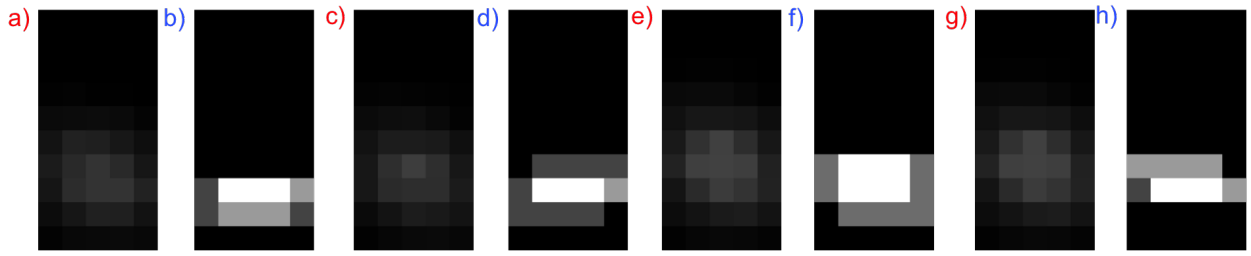

Figure 5.47: *Predicted occupancy of "West3 A" at steps 30, 31, 35, and 36.*

Figure 5.48: *Normalized excluded and overestimate heatmaps for graph "West3 A" for steps 0-15.*

supplement plot in Figure C.19 shows, while $overestimate_s > excluded_s$, $\frac{included_s}{total\_occupancy_s} < \frac{overestimate_s}{total\_predicted_s}$ since $included_s > overestimate_s$ and $total\_occupancy_s < total\_predicted_s$, because there is more prediction swarm in the world in steps 20 through 24 so that $hit\_rate > false\_alarm\_rate$ (Equation 4.18 and Equation 4.19). The result is that while the hit rate is low, the false-alarm rate is even lower, and the sensitivity is above 0.5 and defined.

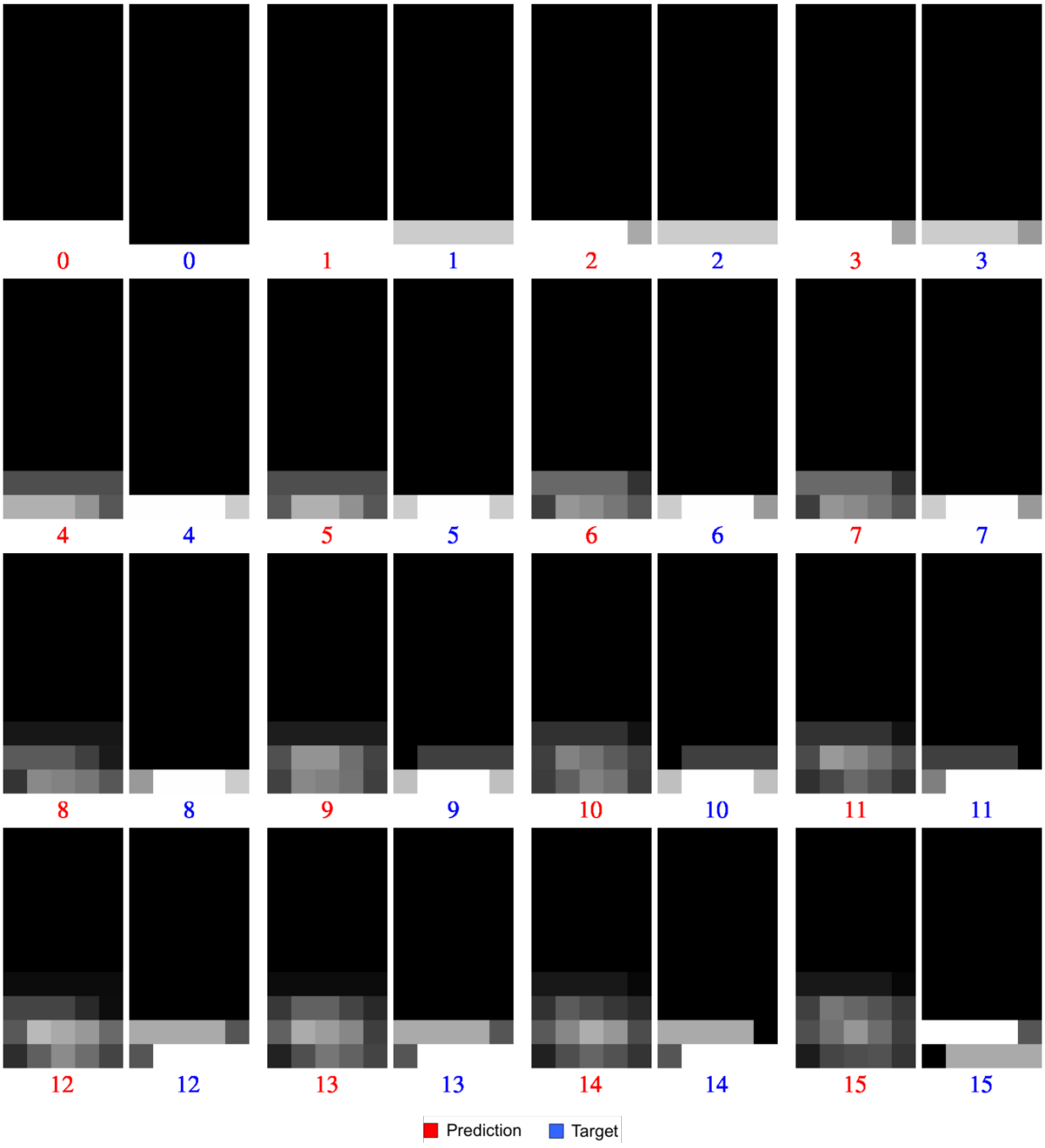

Figure 5.49: *Predicted occupancy for graph "West3 A" for steps 0-15.*

Figure 5.50: *Normalized excluded and overestimate heatmaps for graph "West3 A" for local optima.*

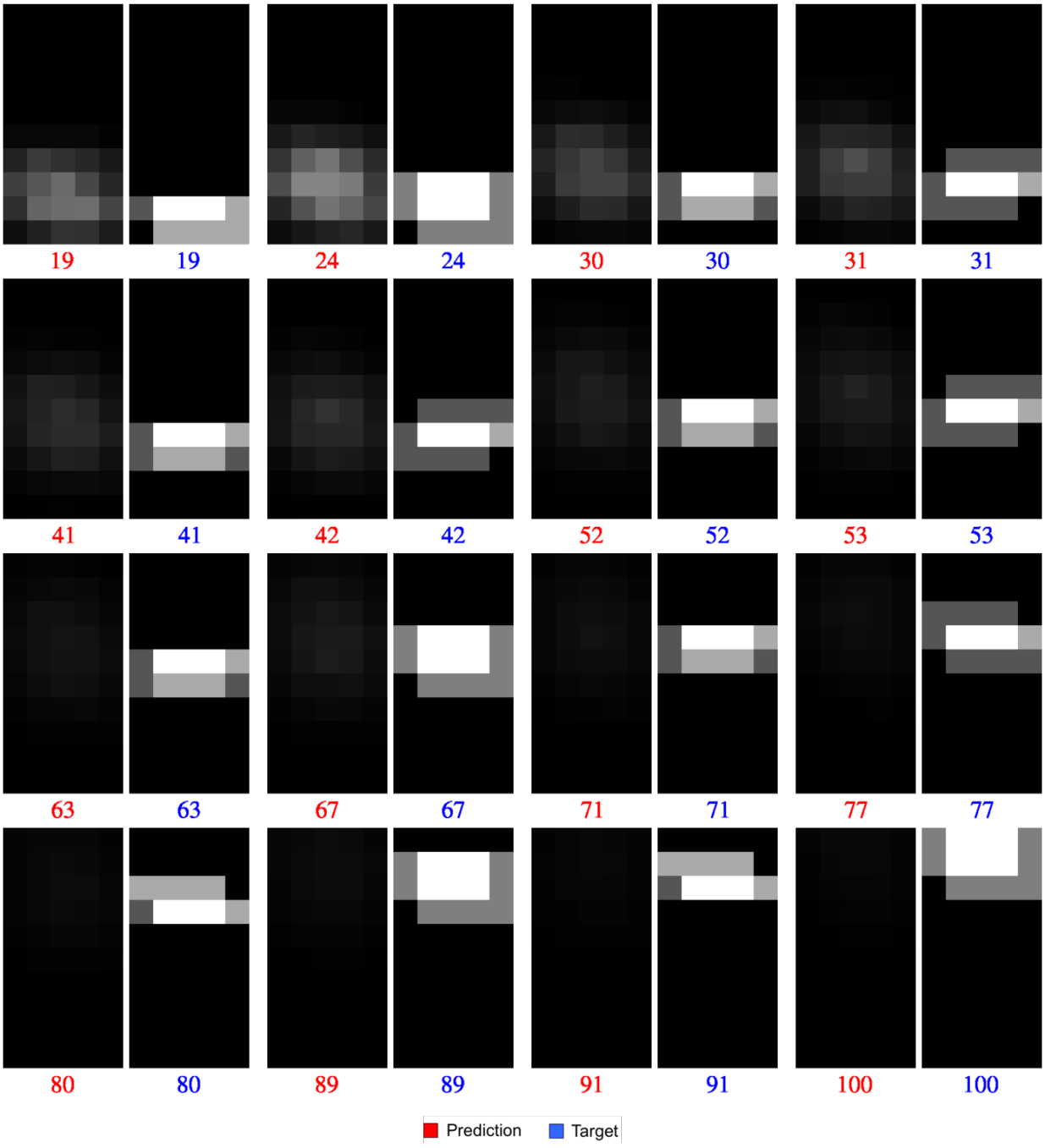

Figure 5.51: *Predicted occupancy for graph "West3 A" for local optima.*

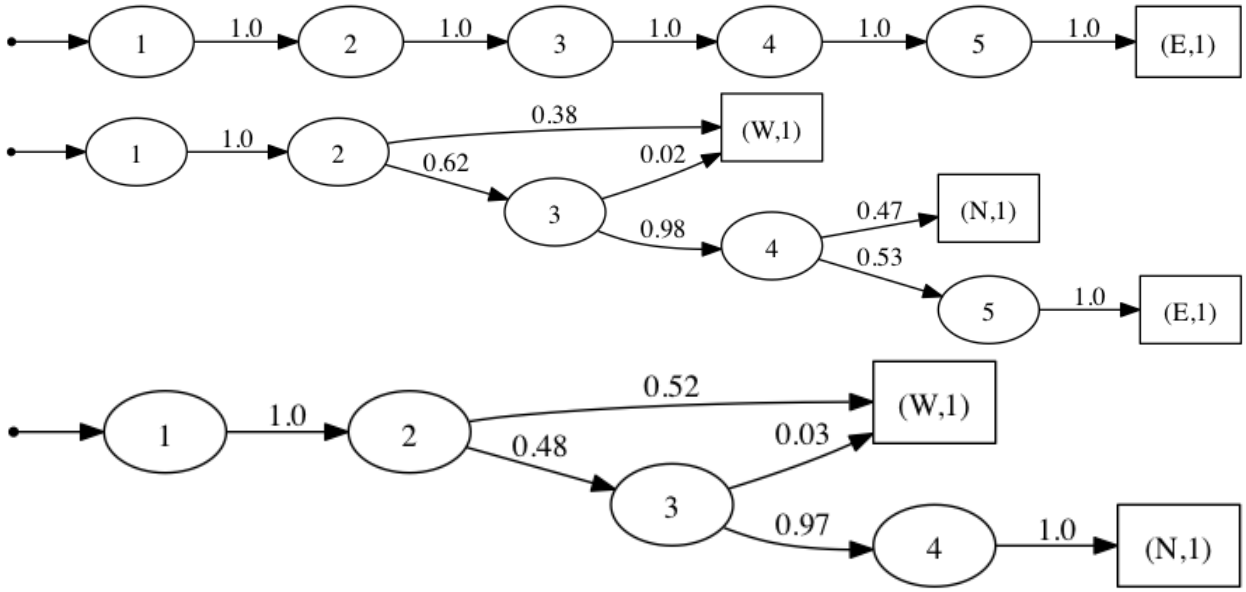

Figure 5.52: *Graphs "West3 B (I)", "West3 B (II-IV)", and "West3 B (V)".*

5.7 Predictions with Diverse Local Behaviors

The poor performance for graph "West3 A" (Figure 5.43) in Section 5.6 seems at odds with the strong repetition observed in the corresponding target occupancy in Figure 5.40. The target swarm distribution looks exactly the same in rows VI and VII in step 70 as it does in rows IV and V in step 50. The swarm also looks very similar in rows II and III in step 30. It seems that this batch execution could be very predictable, yet the occupancy prediction was spread thin (see the occupancy matrices for graph "West3 A" in Figure 5.40) and the prediction value was very poor (see the predictive value for graph "West3 A" Figure 5.44). Upon closer examination, it appears from the heat maps that the behavior in the middle is different from the behavior on the sides. It looks as if there is movement north only in the middle regions but a shift east and west between all of them.

We use the three separate graphs "West3 B (I)", "West3 B (II-IV)", and "West3 B (V)" shown in Figure 5.52 in an attempt to improve the prediction by accounting for the unique

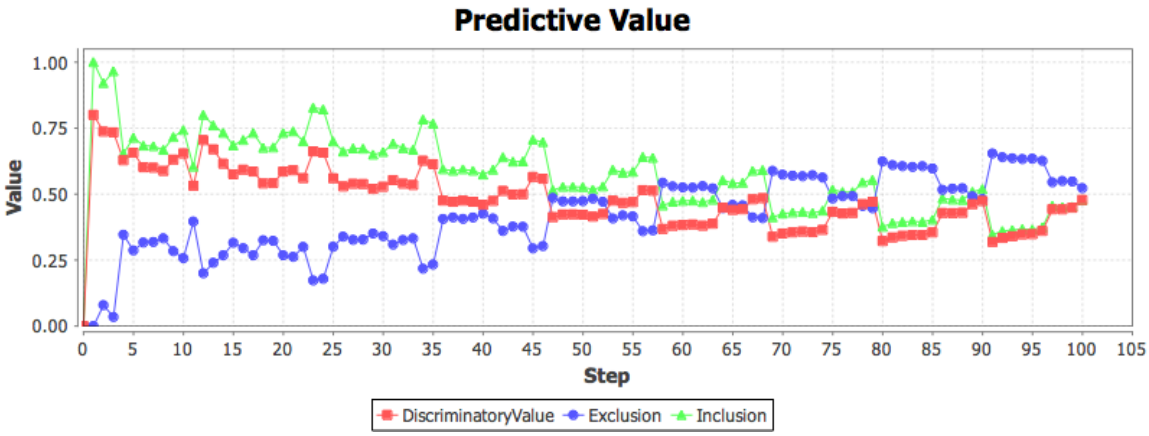

Figure 5.53: *Predictive value for graph "West3 B".*

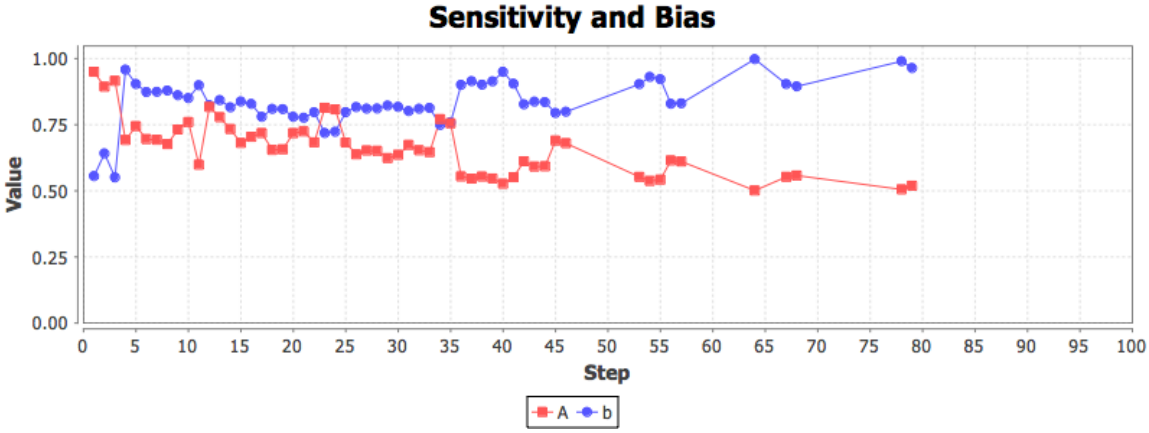

Figure 5.54: *Sensitivity and bias for graph "West3 B".*

behavior observed in columns I, $II - IV$, and V, respectively. The reasoning is that if the behavior along the border regions is very different from the behavior in the middle then attempting to use one graph for all regions might not work very well. Like graph "West3 A" (Figure 5.43) for "West3", the graphs contain few vertices and edges. Furthermore, graph "West3 B (I)" for column I only has an east exit "E" and graph "West3 B (V)" for column V only has west and north exits "W" and "N", respectively. That seems like very specific behavior to attempt to combine into a single graph.

The predictive value for "West3 B (I)", "West3 B (II-IV)", "West3 B (V)" is shown

in Figure 5.53. These three graphs result in better predictive value than the predictive value "West3 A" for plotted in Figure 5.44. It seems as though separating the distinct behaviors in columns I, $II - IV$, and V is beneficial. Similar to graph "West3 A", shifting the steps does not lead to better predictive value or reduced oscillations for these graphs. Of course, there are likely to be oscillations since we are still predicting "West3" where the target swarm oscillates between being spread over two or three rows. Although, the best approach to reducing the oscillations would be to produce a prediction that spreads over the same rows as the target swarm in the same steps. Note that the period in the predictive value is 11, just as it was for graph "West3 A". The fact that the period doesn't change between the predictive value for graph "West3 A" and (see Figure 5.44) graph "West3 B" (see Figure 5.53) suggests that it is heavily influenced by the target swarm in the batch execution. This is not surprising since the paths in the flow diagram in Figure 5.42 for the square pattern (Figure 4.2(b)) show that it takes the target swarm 11 steps to move through a row and the prediction is spread thin and its contribution to $inclusion_s$ is negligable, no different than for "West3 A". More interesting is how the predictive value seems less ordered before step 58 than it does after step 58. This cannot be explained by the predictive value supplement plot in Figure C.19 since $included_s$, $excluded_s$, and $overestimate_s$ show the same behavior there. Instead we look at the the excluded/overestimate and occupancy heat maps for steps before step 58 and steps after step 58. The excluded/overestimate and occupancy heat maps for steps 20 through 35 are shown in Figure 5.57, and Figure 5.58, respectively. The occupancy heat maps show a bright albeit spread and noisy prediction swarm. The excluded/overestimate heat maps look similar. While you can sort of make out where the target swarm is, it is very fuzzy and it changes quite a bit from step to step. The excluded/overestimate and occupancy heat maps for steps 63 through 80 are shown in Figure 5.59 and Figure 5.60, respectively. Contrary to what we saw for steps 20 through 35, the occupancy for steps 63 through 80 shows a darker and hence shallower, more

spread out prediction. The excluded/overestimate heat maps for the later steps show that the prediction swarm is spread so thin that the oscillations of the target swarm between two and three rows has a much more prounounced effect on the predictive value and the movement of the prediction now contributes less of an effect. At the same time, the regional occupancy count plot in Figure C.19 shows that the quickly increasing number of regions exclusive to the prediction swarm (cyan) before step 35 causes $included_s$ and $overestimate_s$ to constantly decrease and increase, respectively, decreasing the dependence of $inclusion_s$ on the number of shared regions. However, after step 35, the more highly spread swarm is more effected by changes in the number of shared regions since the number of regions exclusive to the prediction swarm changes with the number of shared regions, so that $inclusion_s$ is dependent on the number of shared regions.

The sensitivity and bias are shown in Figure 5.54. There we see that b is 1 in step 64, suggesting that there is no bias, yet the predictive value plot in Figure 5.53 shows that $inclusion_s$ is higher than $discriminatory\_value_s$ in step 64. That $inclusion_s$ is higher than $discriminatory\_value_s$ might suggest that $overestimate_s > excluded_s$ and we might interpret that as an overestimate bias. The predictive value supplement plot in Figure C.19 confirms a sizeable gap between the higher $overestimate_s$ and the lower $excluded_s$ in step 64. However, the sensitivity and bias supplement plot in Figure D.13 reveals that the hit rate and false-alarm rate are equal in this step. In other words, while $overestimate_s$ may be larger than $excluded_s$, the ratio of $overestimate_s$ to $total\_predicted_s$ is equal to the ratio of $included_s$ to $total\_occupancy_s$. So, according to b, the prediction is not biased toward either over- or under-estimation (see Figure 4.19).

After seeing the improved predictive value plots it is a little surprising to view the occupancy in Figure 5.55. It isn't the very close looking prediction that might be expected. It also looks noisy like the occupancy for graph "West3 A" (Figure 5.40). Despite this, it has much higher $inclusion_s$ in some of the right places and manages about 50% inclusion

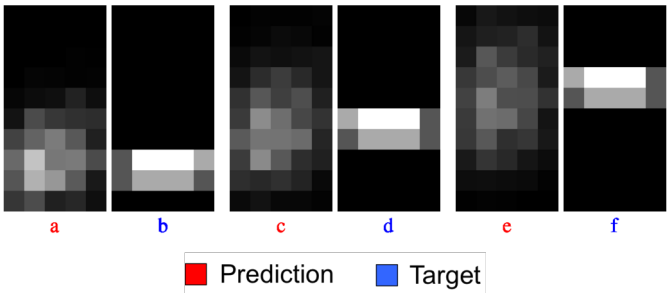

Figure 5.55: *Predicted occupancy for graphs "West3 B (I)", "West3 B (II-IV)", and "West3 B (V)" at steps 30, 50, and 70.*

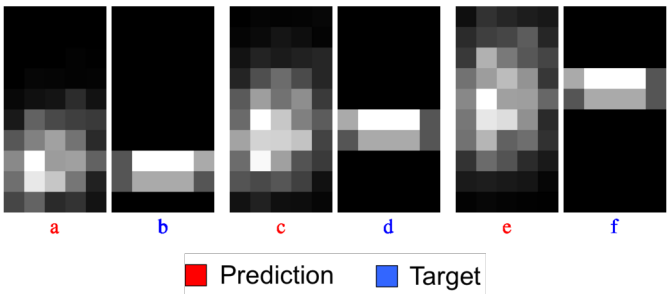

Figure 5.56: *Predicted occupancy for graphs "West3 B (I)", "West3 B (II-IV)", and "West3 B (V)" at steps 30, 50, and 70. Heat maps were adjusted individually.*

Figure 5.57: *Normalized excluded and overestimate heatmaps for graph "West3 B" for steps 20-35.*

and 50% discriminatory value. So we have to ask, why the noisy appearance? Even in Figure 5.56 where the heat maps are adjusted individually, we see a noisy prediction that fails to center on the swarm, at least when viewing the most dense regions in the prediction. Even if we just use the middle of the prediction, it seems to fall behind by step 70, even if it is in a trough in terms of oscillation in the predictive value.

The answer to why the occupancy looks noisy can be found in the flow diagram for the batch execution "West3" in Figure 5.42. In the diagram, we see how the swarm flows through a region in the batch execution. The batch execution is deterministic. In this simple square pattern without targets, it turns out that robots repeat the same simple behaviors again and again, no matter which column they are in. In fact, the differences in activity that we see between the three middle columns and the outer two in the heat maps for "West3"

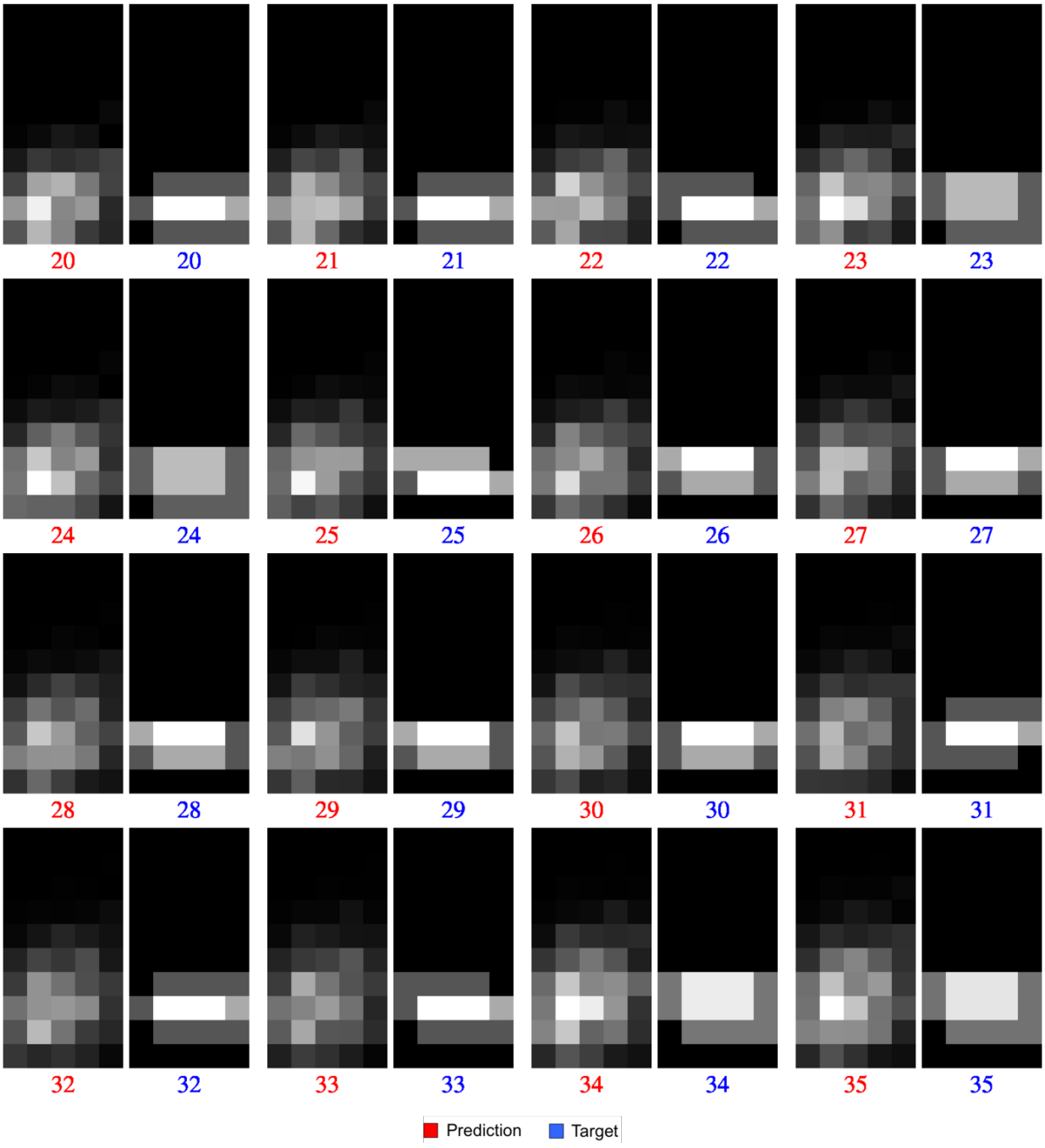

Figure 5.58: *Predicted occupancy for graph "West3 B" for steps 20-35.*

Figure 5.59: *Normalized excluded and overestimate heatmaps for graph "West3 B" for steps 63-80.*

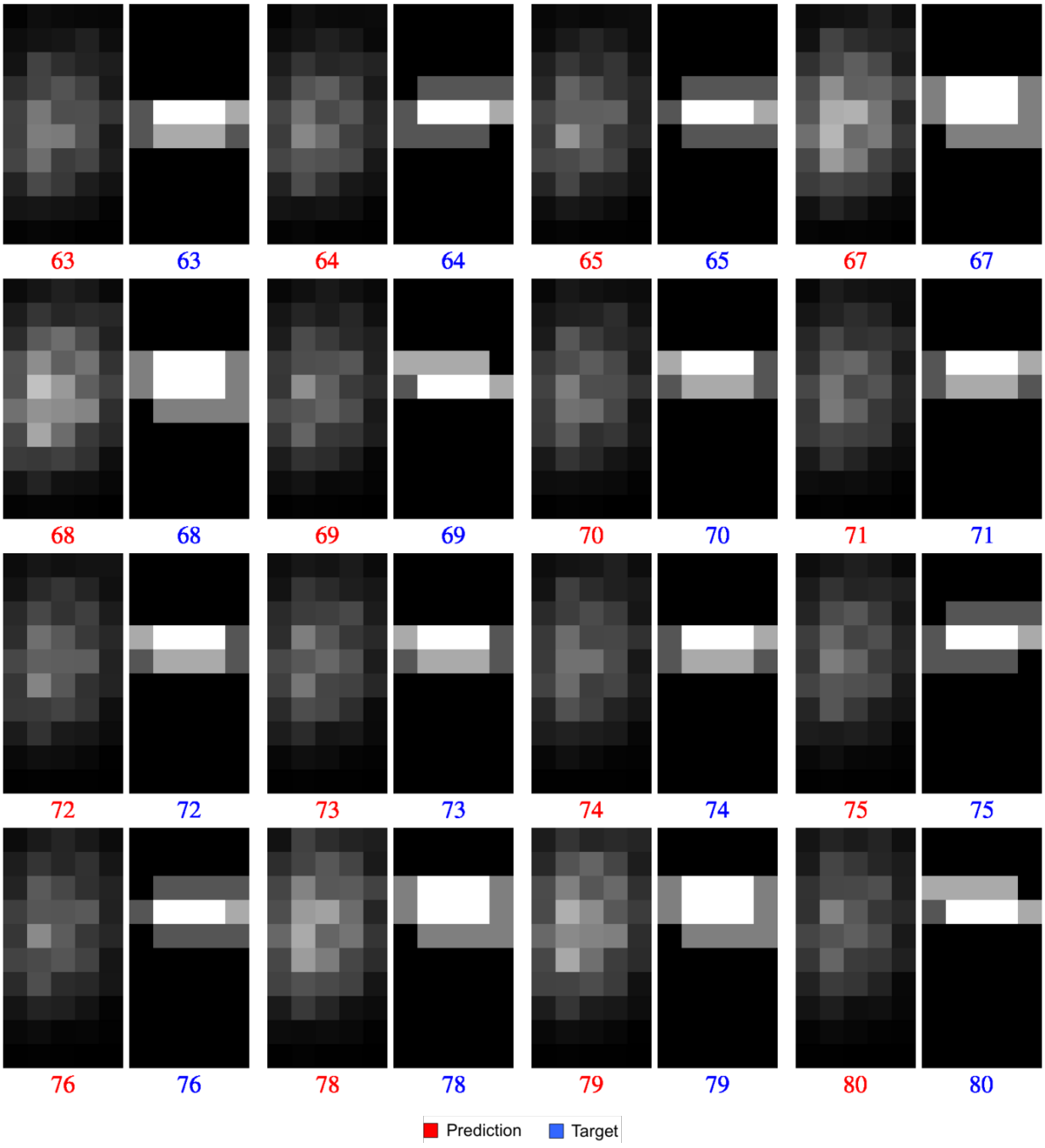

Figure 5.60: *Predicted occupancy for graph "West3 B" for steps 63-80.*

(Figure 5.40) is just a manifestation of these repeated behaviors. It turns out that the batch execution "West1" (Section 5.2) has a similar set of repeated behaviors that are shown in Figure 5.41. Despite this, while using the statistics in Table 5.4 to produce the graph "West1 A" (Figure 5.13) results in the predicted occupancy in Figure 5.12, taking a similar approach and using the statistics in Table 5.6 to produce graph "West3 A" (Figure 5.43) results in the predicted occupancy in Figure 5.40. The predicted occupancy for "West1 A" is close to the target swarm occupancy (Figure 5.12). However, the predicted occupancy for "West3 A" looks very different from the target swarm occupancy (Figure 5.40). In the case of "West1", the robots follow the three paths shown in Figure 5.41 through the regions in "West1". Two of the paths have robots entering from the south and exiting through the north (blue and yellow) and the other path has robots entering from the south and exiting west. One of them is 11 steps long (blue) and another is twelve steps long (green) (the paths are this long because robots either rotate or move forward by one square in a turn but not both). The remaining path (yellow) is 2 steps long. For "West1", it makes sense to combine these paths into the single 12 vertex graph "West1 A" in Figure 5.13. In the graph, about 35% of the robots exit west after 2 steps and the rest essentially exit north, about half after 11 steps, and the remaining after the 12th step, very similar to what we see in the flow diagram. It turns out that the one graph "West1 A" is adaquate to describe these paths through the regions of "West1" because all of the paths enter from the south or east and exit to the west or north.

Unfortunately, it seems that we cannot obtain such accurate predictions for "West3" with one connected graph. It turns out that the paths that robots take through the square pattern in "West3" are more difficult to consolidate. This is because they are completely independent: Following the paths in Figure 5.42, all robots coming from the south exit west, all robots coming from the west exit north, and all robots coming from the east exit east. Figure A.13 shows the 20 robots engaged in this exact behavior in the world execution. If we

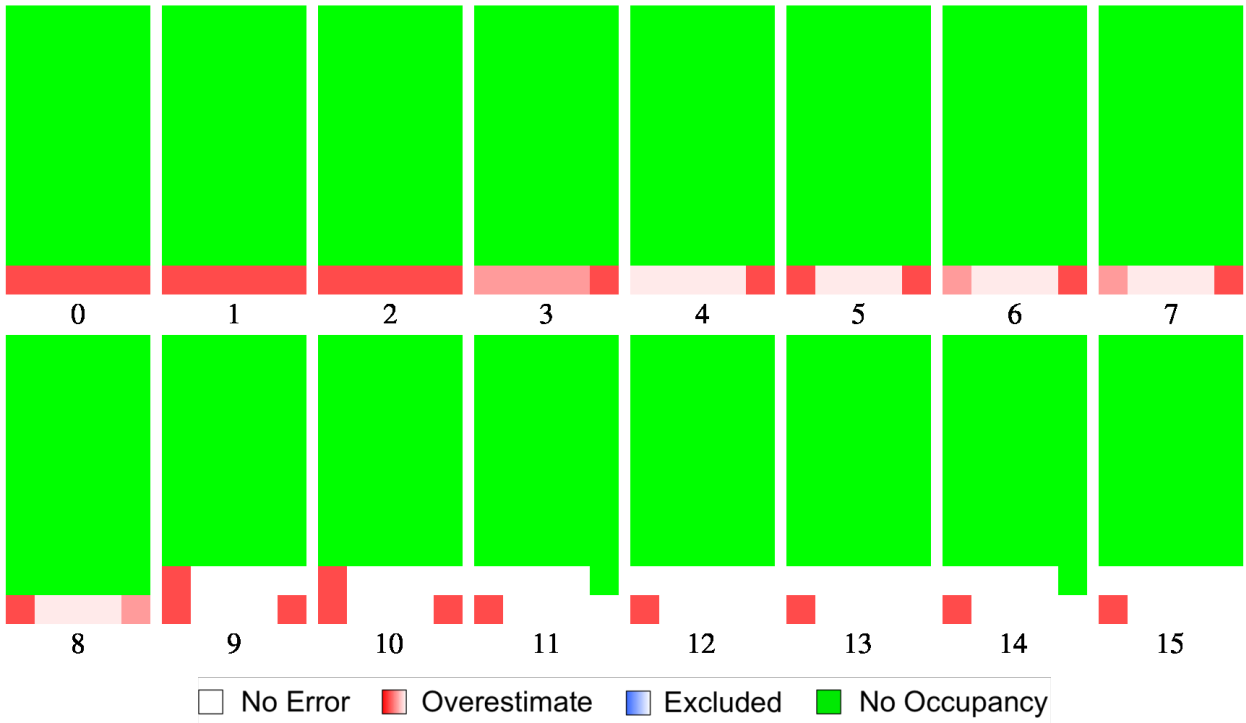

Figure 5.61: *Normalized excluded and overestimate heatmaps for graph "West3 C" for steps 0-15.*

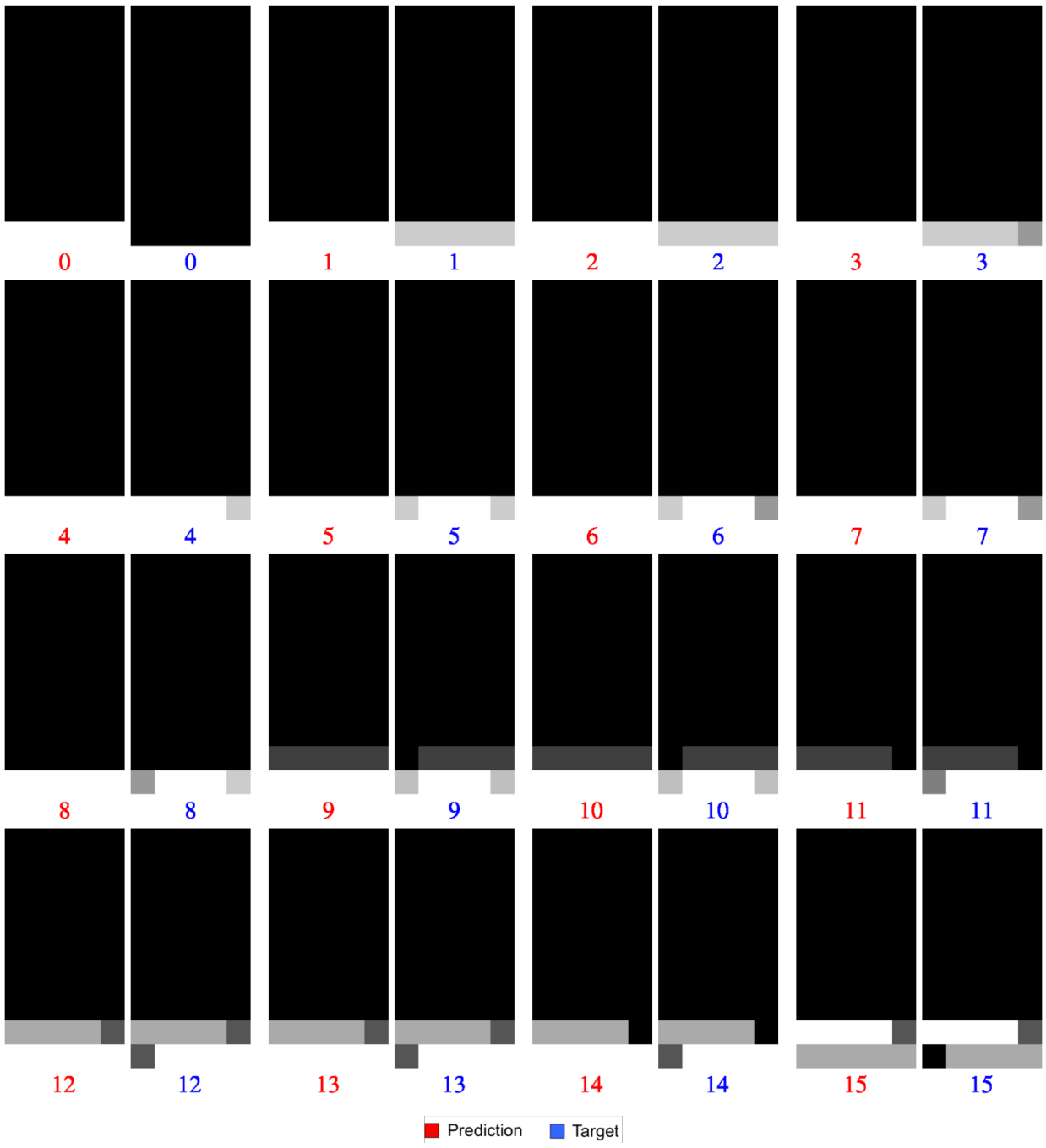

Figure 5.62: *Predicted occupancy for graph "West3 C" for steps 0-15.*

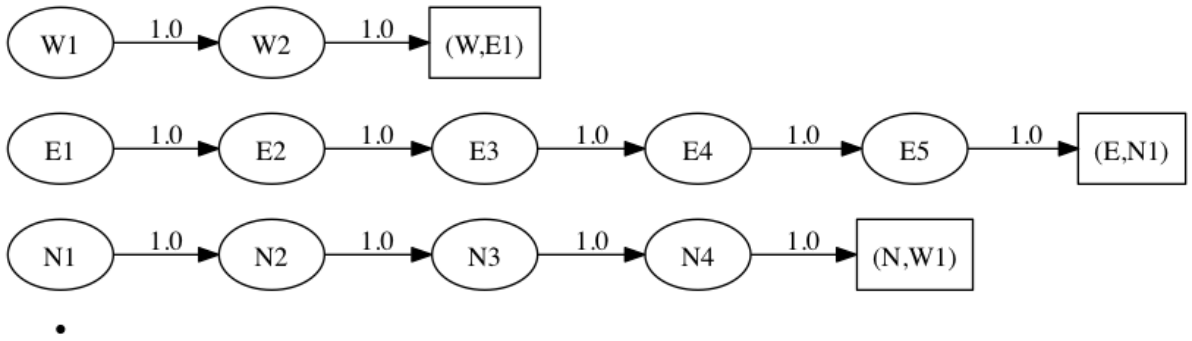

Figure 5.63: *Graph "West3 C" for batch execution "West3".*

try to combine them we end up with a swarm where robots coming in from the west go east and robots that come in from the east go west and so on. This results in movement that is quite noisy and that only roughly matches the behavior of the batch execution. Worse, the lengths of the paths present very distinct behavior. By merging these paths into one graph (see graph "West3 B" in Figure 5.52), we allow some of the robots that enter from the west to exit east in 5 steps and some of the robots that enter in from the west to exit west in 2 steps, and so on. Finally, by blurring the paths that enter from the west, south, and east with those that exit from the west, south, and east in the graph, the robots end up moving back and forth between east-west neighbors. This is something else that does not happen in the batch execution and something that we did not have to worry about for batch execution "West1" where all agents were moving north and west. Consequently, the time the robots spend in the graph before exiting is that much more removed from the batch execution. We show how we can use one separate graph for each of the paths to produce a near perfect prediction in the next section.

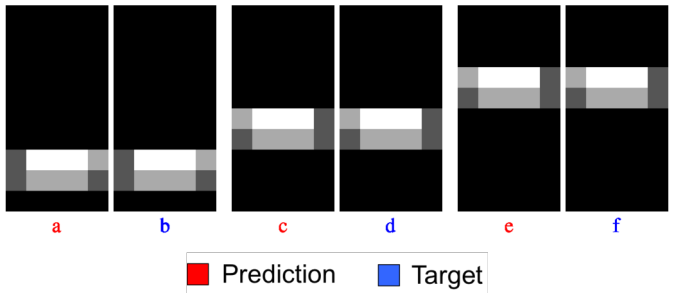

Figure 5.64: *Predicted occupancy for graph "West3 C" at steps 30, 50, and 70.*

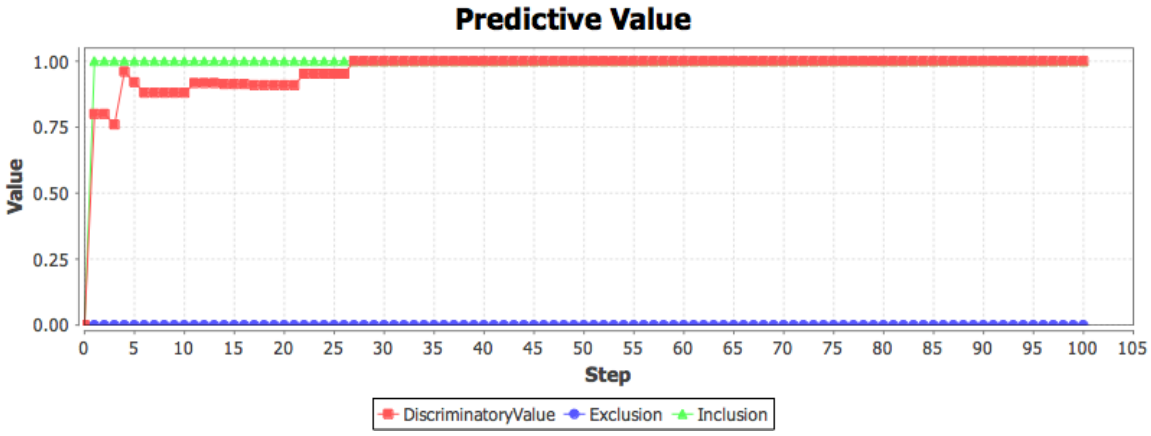

Figure 5.65: *Predictive value for graph "West3 C".*

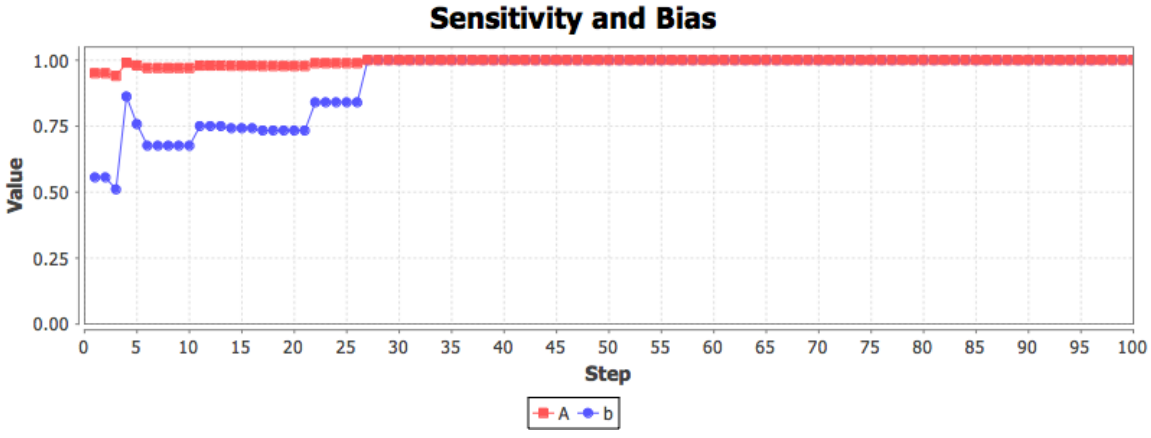

Figure 5.66: *Sensitivity and bias for graph "West3 C".*

5.8 Perfect Predictions using Disconnected Graphs

We can achieve a near perfect prediction for batch execution "West3" by using one graph for each of the three paths shown in the flow diagram in Figure 5.42. We see "West3" in two previous sections. In Section 5.6 we attempt to use a single graph using the method we use to produce graph "West1 A" (Figure 5.13) in Section 5.2 and graph "East" (Figure 5.3) in Section 5.1. That is, we use a table of statistics summarizing the proportion of robots to exit in any direction (north, south, east, or west) after occupying the square pattern for a number of steps to produce a graph where same proportions of robots exit in the same directions after the same number of steps. In Section 5.7 we attempt to account for what looks like the unique behaviors in the left, right, and middle columns by using one graph for column I, one graph for column V, and another graph for columns $II - IV$. In both cases we end up with less than ideal occupancy and predictive value due to the three independent paths that the robots take through the regions shown in the flow diagram for batch execution "West3" in Figure 5.42. The best solution is to produce a graph with three disconnected sub-graphs, one for each of the paths, similar to the two part graph that we use for graph "West2 B" (see Figure 5.34) in Section 5.5 where an impassible barrier divides each region into two disconnected zones, each unreachable from the other. Graph "West3 C" is shown in Figure 5.63. The occupancy is shown in Figure 5.64, the predictive value is shown in Figure 5.65, and the sensitivity and bias is shown in Figure 5.66. But we only use graph "West3 C" for rows II-X due to the tumult common in row I at the beginning a world execution. We show how we use another graph for row I in the next section. By using a different graph for row I, we achieve perfect predictive value and sensitivity and bias for all steps after 26. There is a fair amount of overprediction in the steps leading up to step 26. The high $overestimate_s$, decreased $discriminatory\_value_s$, and overestimate bias can all be explained by the excluded/overestimate and occupancy heat maps for the first

16 steps in Figure 5.61 and Figure 5.62. In the excluded/overestimate heat maps there is only $overestimate_s$ (red) among perfect predictions because, as the occupancy heat maps show, the prediction swarm and target swarm are not doing the same thing in row I. There is only $overestimate_s$ because the prediction occupancy is always at least as high as the target occupancy in the occupied regions. The prediction and target swarms are, however, doing the exact same thing in row II in the occupancy heat maps, and this continues for rows $III - X$. The fact that the swarms overlap prefectly after step 26 can also be seen in the excluded/overestimate heat maps in Figure C.24, the predictive value supplement plots in Figure C.22, and the sensitivity and bias supplement plots in Figure D.15.

5.9 Predictions and Initial Conditions

While graph "West3 C" (Figure 5.63) in Section 5.8 does describe the exact behavior that we observe in rows II-X of "West3", we introduce another graph to mitigate the effect of the robot collisions in row I. The first 26 steps of "West3" (including step 0) are shown in Figures B.1 - B.5. The robots are color coded for ease of tracking individually from step to step.

In the figures, we see that it takes the swarm a while to line up into the paths that we observe in the upper rows. In columns II-V, the robots leave the first row at steps 9, 12, 15, 20, and 25. They start out lined up horizontally at the bottom of the world (Figure B.1) and have to avoid obstacles and each other before they eventually end up in row II (corresponding to vertex "W1" in graph "West3 C" in Figure 5.63, the only vertex accessible from the south, consistent with the paths in the flow diagram for "West3" in Figure 5.42). Once in row II, the pattern of behavior is the same in rows II-X for the rest of the batch execution as the robots follow the paths in Figure 5.42 through the regions.

The behavior in column I is similar to that of the other columns, but unique due to the

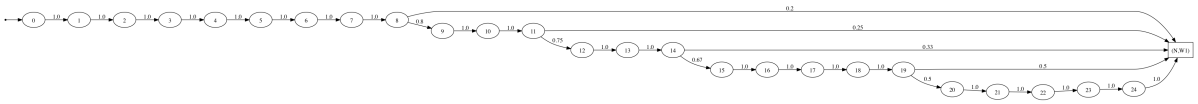

Figure 5.67: *Graph "West3 C (I)" for batch execution "West3".*

fact that the robots are entering the regions from the squares on the west edge of the world. The robots happen to be headed toward the west edge, while the move away from the east edge. The behavior between column I and the other 4 diverges in step 4 when the black robot leaves the world rather than turning northward in a lateral phototaxis move (a robot will exit once it is in the outer ring of squares facing outward). The red robot soon follows the black robot out of the world (Figure B.2) and the yellow and blue robots exit from row II and have left the world by step 18 (Figure B.4).

The behavior in columns II-V is the same, and even despite the robot-robot interaction. The behavior of the black robot does not change from step 0 (Figure B.1). The green robot to the west of each region (the green robot in column V exits at step 5 (Section B.2)) and the red robot begin their repetative behavior at steps 2 and 3 (Section B.1), respectively. The green robot from the west region takes the lead and leaves the region first at step 9 (Figure B.2). Black follows after green and exits the region at step 12 (Figure B.3). Red starts out in collision with an obstacle and has to avoid it before getting on track but eventually follows black and exits at step 15 (Figure B.4). Yellow and blue take a route that is not used in the other rows. Yellow collides with black and blue with yellow in step 9 (Figure B.2) and yellow and blue spend several steps deviating from goal directed behavior before phototaxis causes them to turn around and follow the others in steps 12 and 14 (Figure B.3), respectively. Yellow ends up exiting at step 24 (Figure B.5) and blue at step 26 (Figure B.5).

With this information available, we ignore the collisions and simply take advantage of the fact that the robots in each column leave at exact steps. We use the graph "West3 C (I)" in

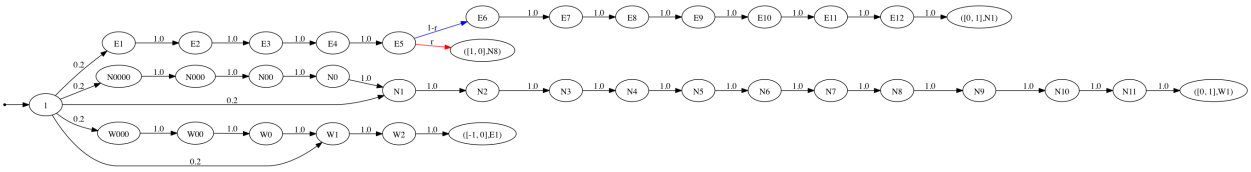

Figure 5.68: *The graph template for row I for batch execution "West4".*

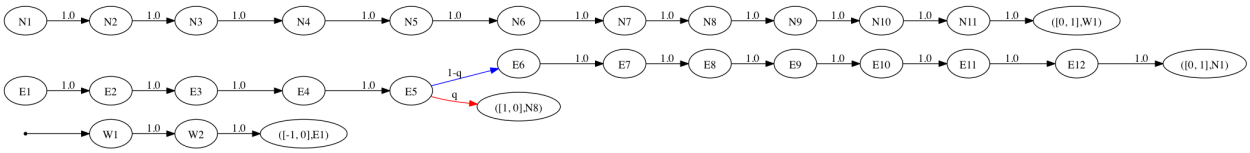

Figure 5.69: *The graph template for rows II − X for batch execution "West4".*

Figure 5.67 for the regions in row I rather than "West3 C". This graph ensures that robots enter row II in the exact vertex and step so that we can demonstrate perfect predictive value for the graph "West3 C" (Figure 5.63) in the steps following 25. Note that this leads to less than perfect predictive value in the first row. We could use a larger graph to model the exact behavior of the first row if we want perfect behavior for the duration of the batch execution. Together the graphs "West3 A" (Figure 5.43), "West3 B" (Figure 5.52), and "West3 C" (Figure 5.63) demonstrate that the same batch execution can be modeled with different graphs to achieve different purposes. A larger graph might be used to model the exact behavior of a deterministic swarm or a simpler one might be used to model the general direction and rate of movement, with some loss of information about the exact regions that the robots might be in. This example also suggests that using a different graph for the first row might sometimes be beneficial.

5.10 Predicting in the Presence of Non-Determinism

We have seen two different behaviors for the square pattern "West1", using the two different values of 1.0 and 0.0 for `rtp` for the batch executions "West1" (Section 5.2) and "West3"

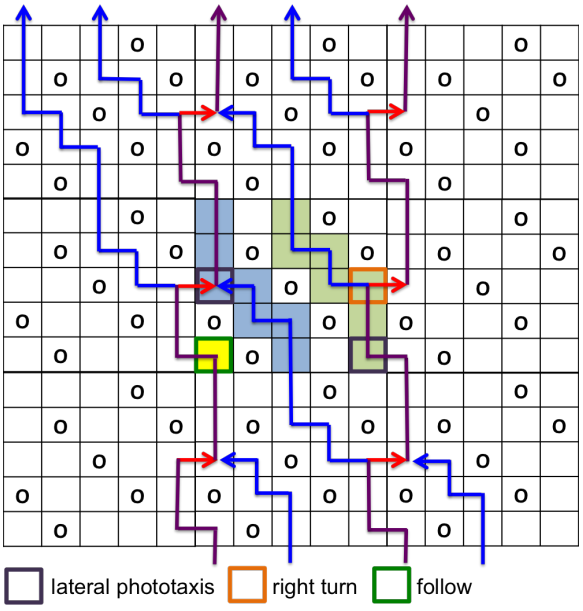

Figure 5.70: *The local behavior of batch execution "West4".*

(Section 5.6), respectively. Now we predict what happens when `rtp` is 0.5. In batch execution "West1", the `rtp` of 1.0 causes the target swarm to move north-west (see the flow diagram for "West1" in Figure 5.41). In batch execution "West3", the `rtp` of 0.0 causes the target swarm to move north (see the flow diagram for "West3" in Figure 5.42). In batch execution "West4", the target swarm moves north-north-west since `rtp` is 0.5. This is a path somewhere between the other two. That is because robots move north half of the time and west half of the time. The proposed behavior is demonstrated in in the flow diagarm in Figure 5.70. This flow is a hybrid of the corresponding "West1" and "West3" flows.

Of course, it does not have to be so simple. Collisions originating at the points where the paths converge could lead to behavior that is more difficult to predict. But with each of those collisions phototaxis (`pp` is 1.0) will ensure that, even though robots might be temporarily disoriented, they ultimately turn around and continue north or north-west.

With all of these collisions going on we can expect the robots to be spread over a few rows. We can also expect that the robots in one local area will be arranged very differently

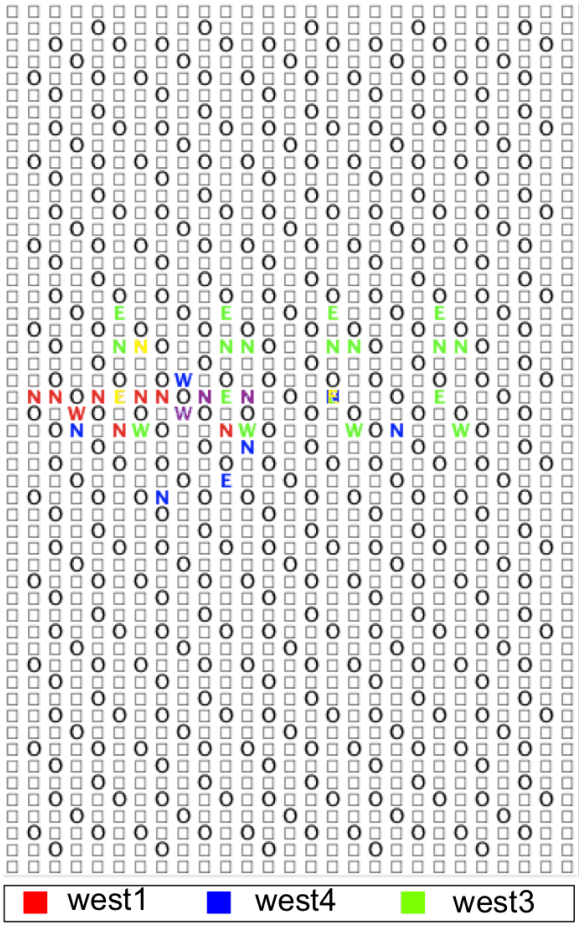

Figure 5.71: *Batch executions "West1", "West3", and "West4" merged at step 70.*

from those in another area unlike what we see in deterministic batch executions in previous sections. Figure 5.71 shows three superimposed screen shots from step 70 of the "West1", "West3", and "West4" batch executions. In the figure, the robots of "West4" end up somewhere between the robots of "West1" and "West3". Purple designates the overlap between red and blue and yellow designates the overlap between green and blue.

The graph templates "West4 I" and "West4 II-X" for batch execution "West4" are shown in Figure 5.68 and Figure 5.69, respectively. We use templates for batch execution "West4" because it is a non-deterministic batch execution and we produce more than one graph, each for a different sample size (see page 53 in Section 4.4.3). We use the templates to produce the graphs for the different sample sizes. Each template is parameterize by one probability. The template "West4 II-X" is the template for row $II - X$ and is parameterized by the probability q. The template "West4 I" is the template for row I and is parameterized by the probability r. We use a separate graph for row I to avoid allowing behavior that is unique to row I that occurs as robots first enter the world to influence the graph for rows $II - X$. We used a separate graph for row I in Section 5.9 when we used graph "West3" in Figure 5.63 and "West3 (I)" in Figure 5.67 to predict batch execution "West3". For each sample size, we use two sets of exit statistics (Section 4.4.3), one set for row I and one set for rows $II - X$. Each set of statistics is caculated over a sample of the data set for the batch execution and only for activity in the rows of interest. Each set of statistics describes, for each pair of adjacent neighboring squares u and v, the proportion of robots that enter from u to exit to into v after occupying the region for s steps. From the exit statistics for rows $II - X$ we obtain the value for q and from the exit statistics for row I we obtain the value for r. Specifically, we obtain the proportion that enter from $E(0, 0)$ to exit into $E(0, 2)$ after 5 steps. That is, we are interested in robots that enter from square $(0, 0)$ in the region to the east and exit into square $(0, 2)$ in the region to the east 6 steps later. The values of q and r can be found in Table 5.7. Notice that template "West4 II-X" in Figure 5.69 closely

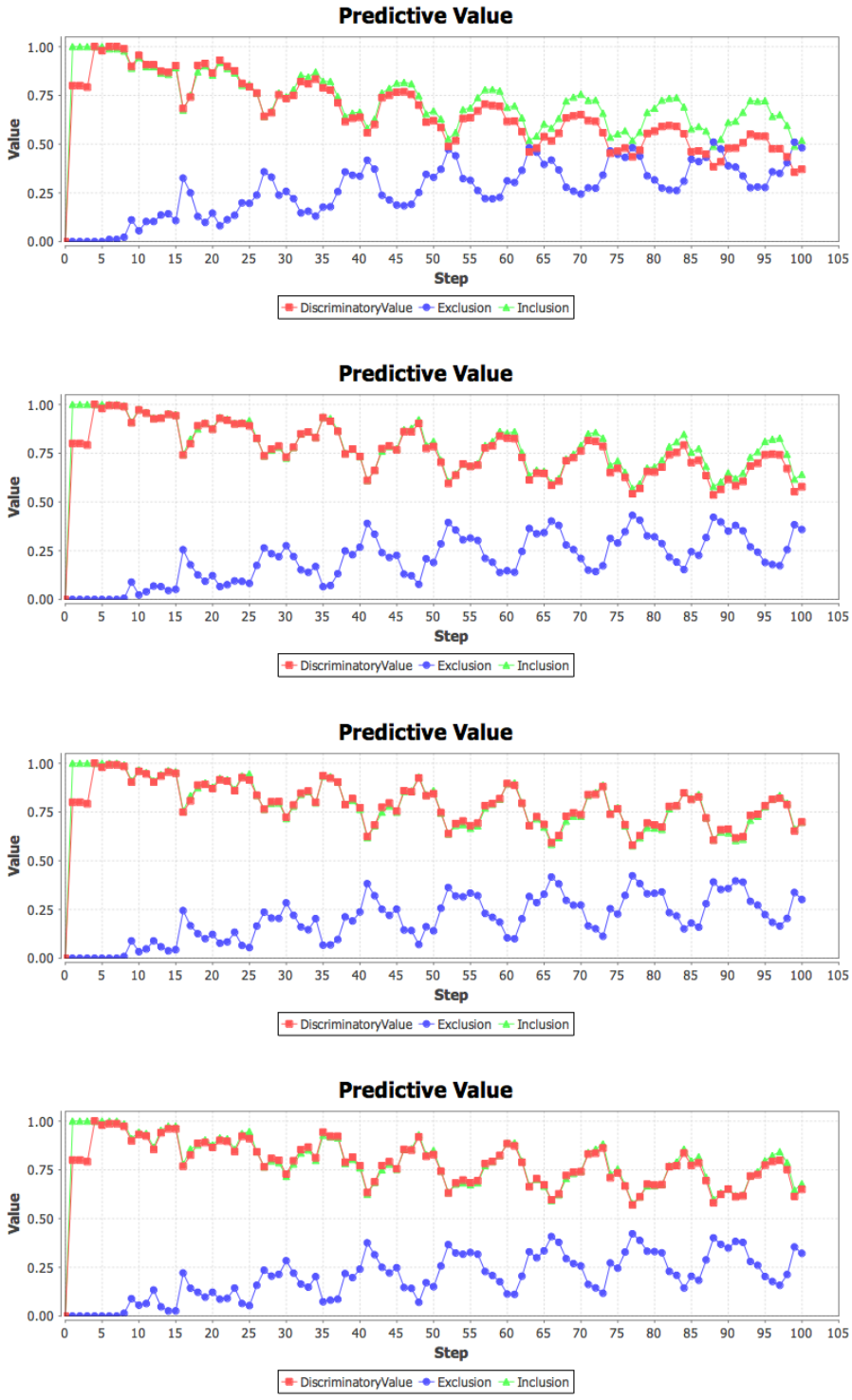

Figure 5.72: *Predictive value for graph "West4".*

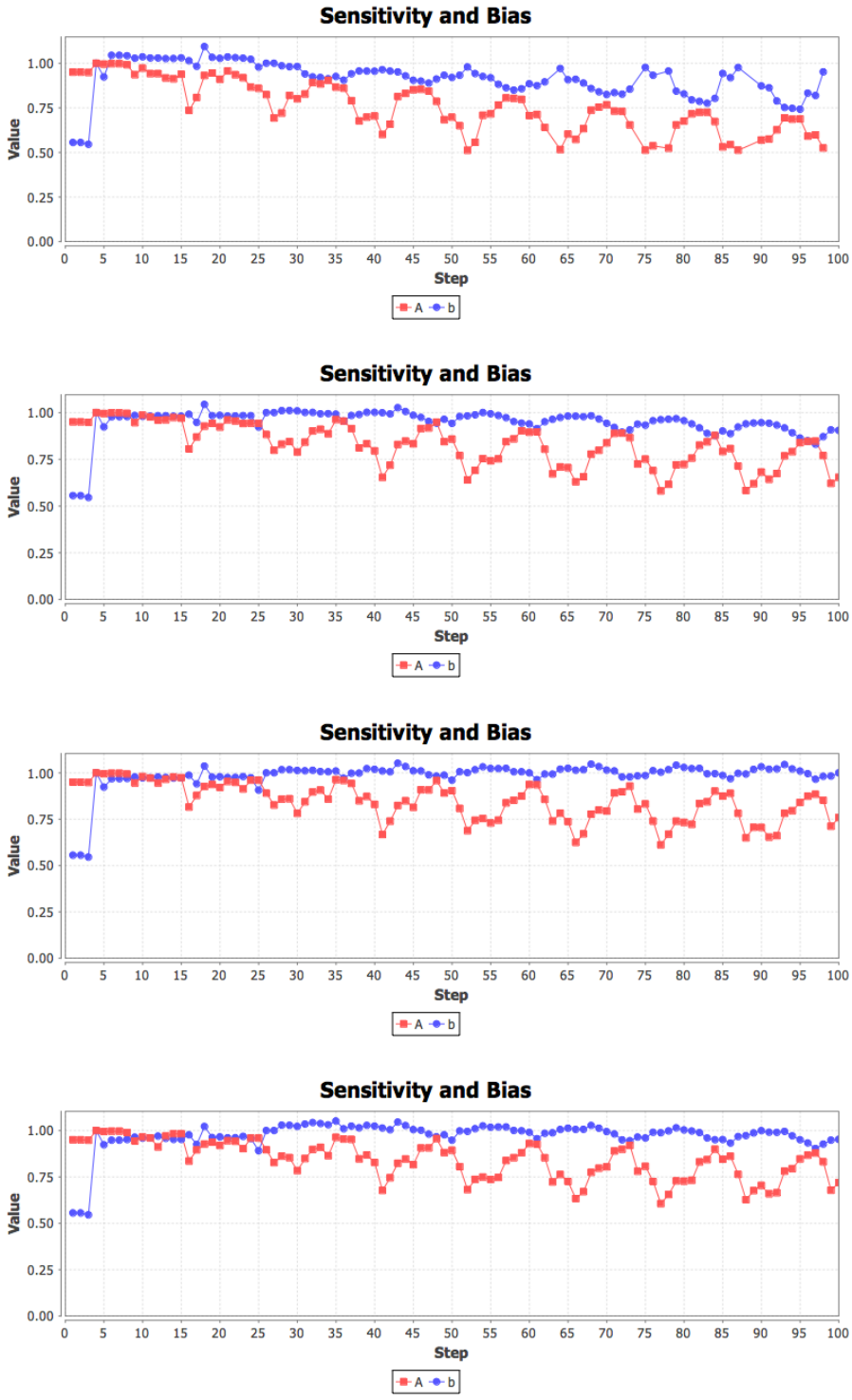

Figure 5.73: *Sensitivity and bias for graph "West4".*

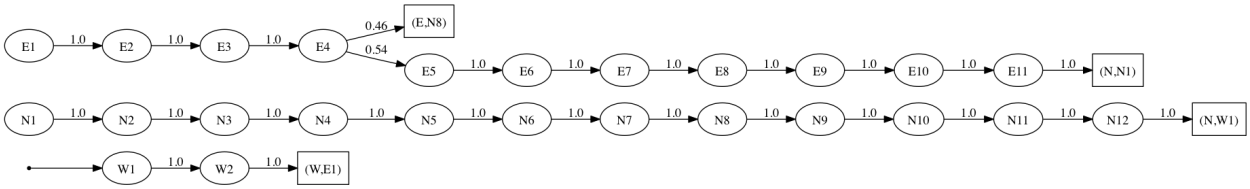

Figure 5.74: *Graph "West4" for batch execution "West4".*

reflects the swarm flow in the flow diagram in Figure 5.70. Each subgraph of "West4 II-X" is associated with a path in the diagram. From vertex "E5", robots will decide whether to follow the red edge into "N8" with probability q or follow the blue edge into "E6" with probability $1-q$. The vertex "E5" in the template corresponds to square $(4, 2)$ in the region in the diagram. Following the red edge corresponds to moving into $E(0, 2)$ and following the blue edge corresponds to moving into $(3, 2)$. While `rtp` is 0.5 for the batch execution, q is not necessarily 0.5 because we ran the statistics on the exits, like we did for other batch executions, to come up with the numbers. They account for robot interference, such as collisions, that result from the uncertainty in turn direction. In addition, a robot in an adjacent square can cause another robot to use the follow move where it would have used a normal turn, by obstructing the robot on one side. In general, robots can interfere with the phototaxis and tagging of another as well. In this batch execution, the non-determinism due to `rtp` ensures that these events continue indefinitely.

Table 5.7: *Values of q and r by sample size.*

Sample Size	**# Paths**	q	r
1	25	0.6821	0.7500
2	50	0.5284	0.3700
4	100	0.4453	0.3125
8	200	0.4876	0.1875

We use sample sizes of 1, 2, 4, and 8 world executions. This translates into 25, 50,

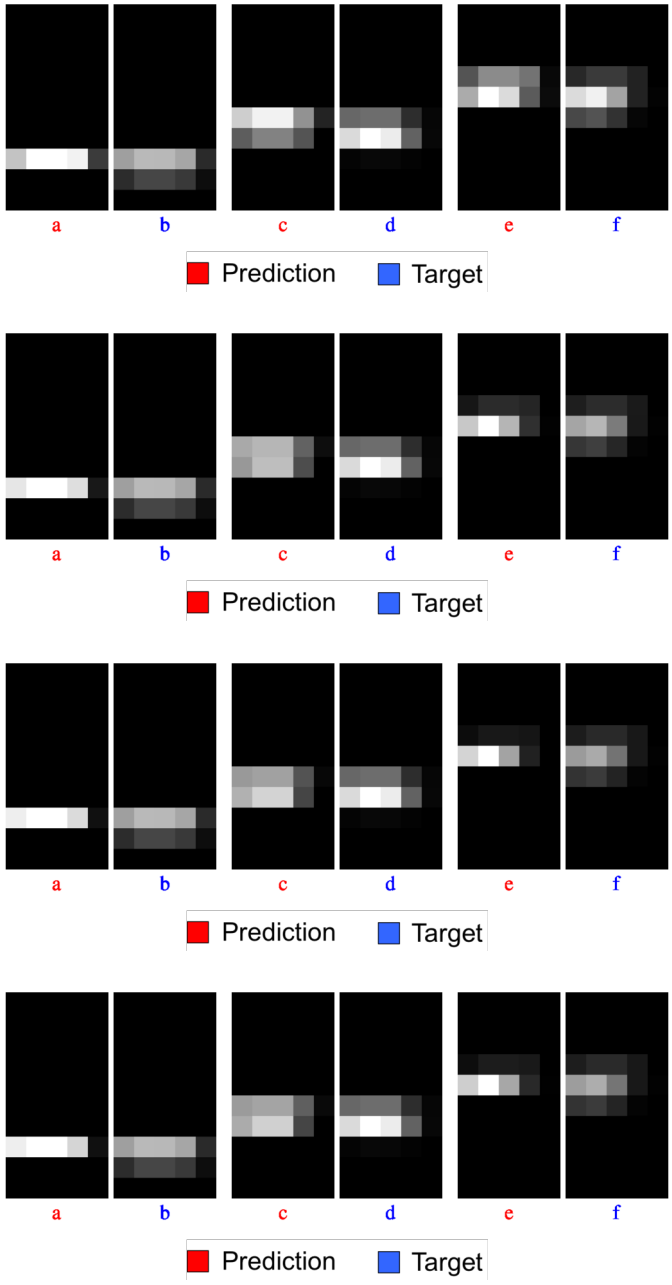

Figure 5.75: *Predicted occupancy of graph "West4" at steps 30, 50, and 70.*

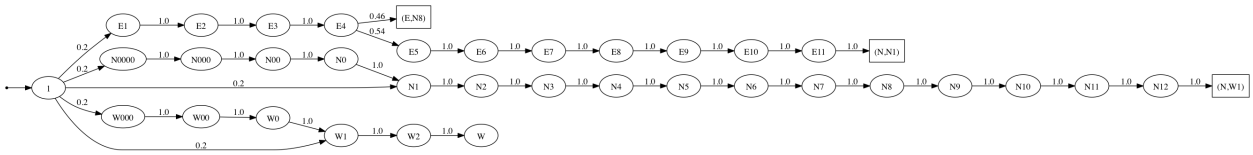

Figure 5.76: *Graph "West4 (I)" for batch execution "West4".*

100, and 200 robot paths from the data set. We use the number of world executions to keep the data for world executions atomic. This only means that we use $25k$ paths from the data set where k is the chosen number of world executions. It does not mean that we alter the world executions or the data set in any way. The multiplier is 25 because there are 25 robots in a small world and, hence, 25 paths are generated for a small world. Using a number of paths other than l, such that $l\%25 = 0$, might not be representative of the behavior in the batch execution because it would use partial data from a world execution. We stop at a sample size of 8 because we run into diminishing returns after a sample size of 4 or 8. The occupancy matrices for the resulting four graphs are shown in Figure 5.75 with the occupancy for a sample size of 1 at the top and a size of 8 at the bottom. The occupancy matrices show small differences between sample sizes, but the occupancy for 2 and 8 looks close. The predictive value is shown in Figure 5.72. The $inclusion_s$ and $discriminatory\_value_s$ are lower for a sample size of 1. The $discriminatory\_value_s$ is lower for a sample size of 2. There is a slight drop in $discriminatory\_value_s$ for the sample size of 8. The excluded and overestimate heat maps in Figure 5.83 and occupancy heat maps in Figure 5.84 follow the predictive value from one trough to another from step 75 to step 90 for the sample size of 4. They show a complex and gradual movement in both the prediction and target swarm unlike the oscillations in the previous sections that are much more course. These movements, combined with the fact that the prediction and target swarm are out of phase, result in the oscillations in the predictive value. The period associated with these oscillations appears to be about 11. This may be related to the fact that the blue and

green paths in the flow diagram for batch execution "West4" in Figure 5.70 are 11 and 12, respectively. Some of the prediction also moves through the world in 11 or 12 steps (see the paths exiting through virtual vertices "([0,1],W1)" and "([0,1],N1)"in the graph template shown in Figure 5.69). The gradual, out-of-phase, movement of the prediction and target swarm for the data set with 100 paths can be more easily seen with the mutually adjusted excluded and overestimate heat maps for steps 75-90 in Figure 5.78. The mutually adjusted heat maps are normalized by dividing all values by the largest value across all heat maps in the range (steps 70-90) so that relative differences in their values can be compared. The mutually adjusted included heat maps for the data set with 100 paths and steps 75-90 can be found in Figure 5.79. The sensitivity and bias are shown in Figure 5.73. The values for the sample size of 8 seem more like the values of the sample size for 2. The fact that the q increases from the previous sample size may contribute to this. The steps for this batch execution are shifted by 1.

5.11 Evolutionary Graphs

Having now had the opportunity to view and compare the prediction occupancy matrices and predictive value of several graphs, we are in a better position to discuss the trade-offs between graphs of different sizes and complexity. As we have mentioned previously, there can be a trade-off between the larger graphs with the higher predictive value and smaller graphs with lower predictive value. But we have also mentioned that in at least some ways the larger graphs might be just as intuitive, if not more, than the smaller graphs.

Take, for example, the graph "West1 B" in Figure 5.85. It is an evolutionary graph predicting the target occupancy matrices of batch execution "West1". Recall that we use path length histograms (Section 4.3.3) to evolve graphs (Section 4.4.3). A comparison of the prediction and target histograms for columns I and III is plotted in Figures 4.11 and 4.12.

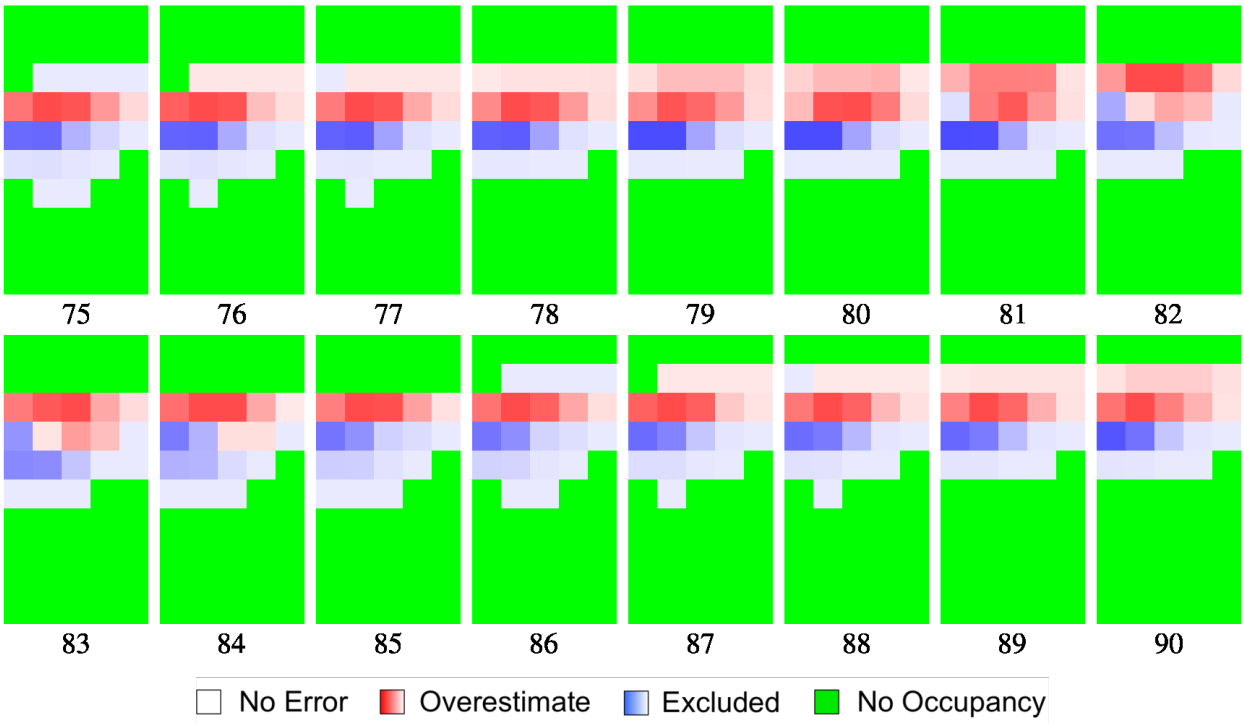

Figure 5.77: *Normalized excluded and overestimate heatmaps for graph "West4" for steps 75-90 and sample size of 1.*

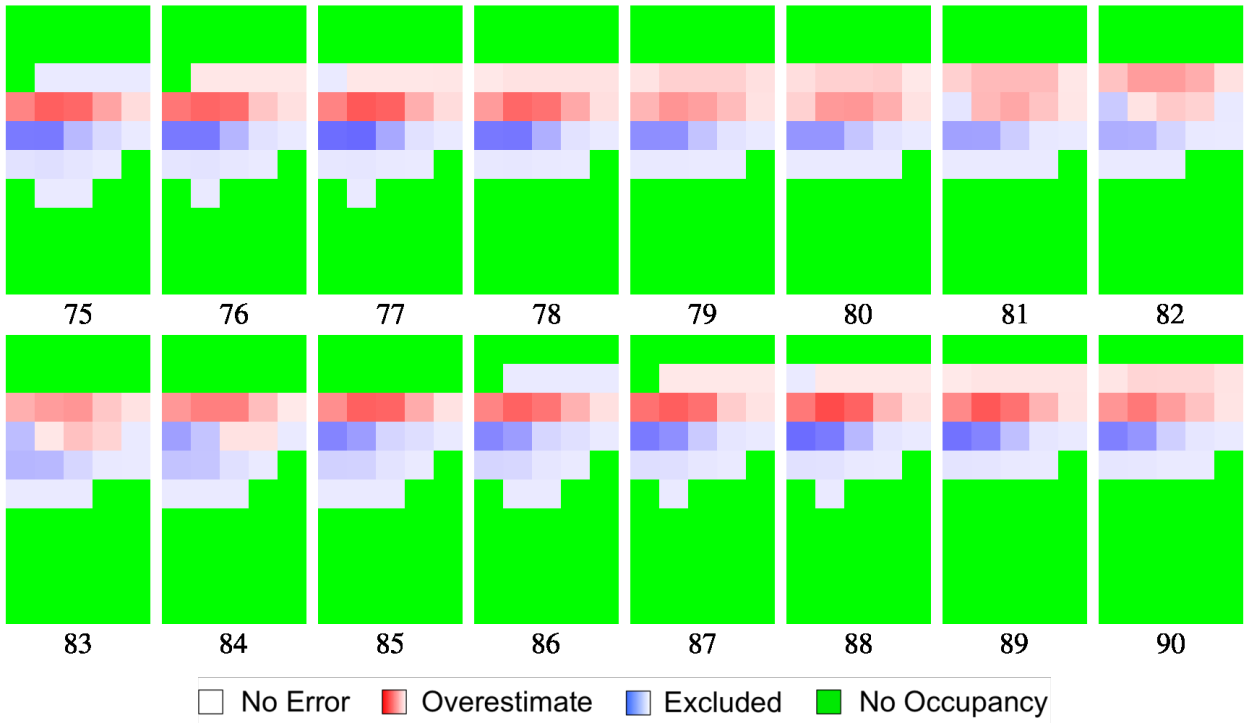

Figure 5.78: *Mutually normalized excluded and overestimate heatmaps for graph "West4" for steps 75-90 and sample size of 1.*

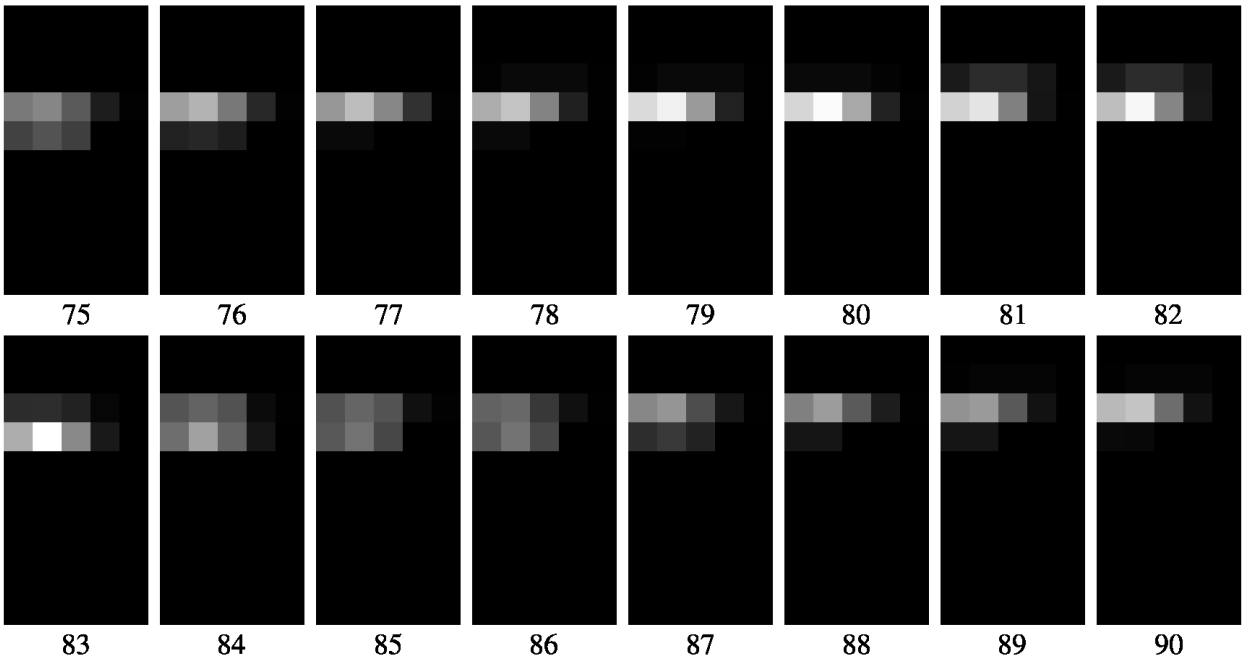

Figure 5.79: *Mutually normalized included heatmaps for graph "West4" for steps 75-90 and sample size of 1.*

Notice that this graph is quite a bit more concise than some of the others, at least in the number of vertices. You might also notice that, unlike any that we have seen so far (at least for real vertices), this graph is cyclic. It is possible that we could do without vertex "s" and "(S,n)" since the robots are headed north-west, even if we have to adjust the probabilities a bit. That would result in an even more concise graph. Nevertheless, the graph captures the general rate at which the swarm moves north and west. While it doesn't do as good of a job of determining the exact proportions in regions, the heat maps suggest that it does correctly predict where the occupancy is highest.

The predicted occupancy is shown in mutually adjusted heat maps in Figure 5.86, meaning that these heat maps are adjusted together making them comparable (Section 4.3.2). They indicate that the proportions predicted are much lower than the actual. However, the individually adjusted heat maps in Figure 5.87 show a slightly different story. These heat

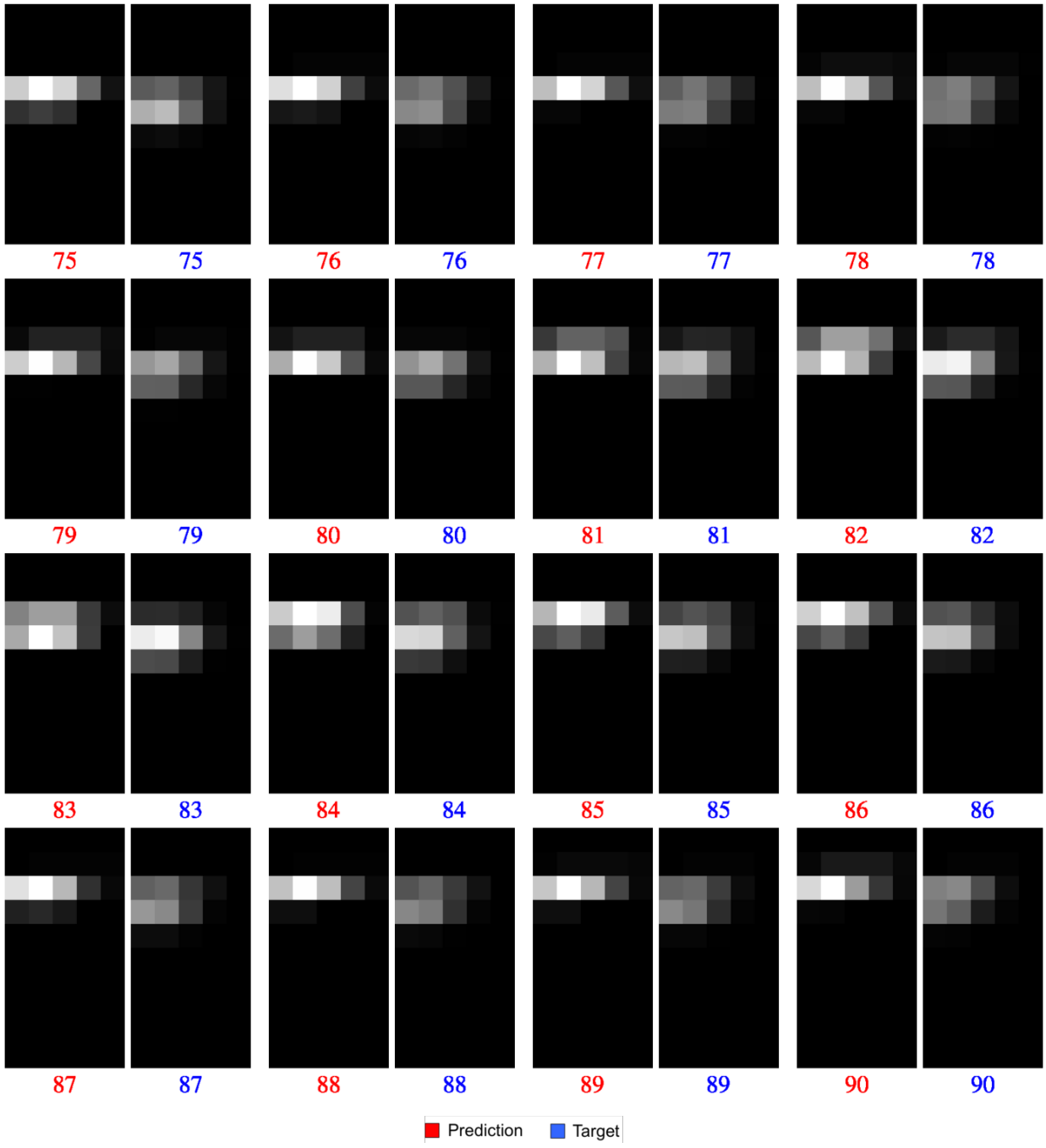

Figure 5.80: *Predicted occupancy for graph "West4" for steps 75-90 and sample size of 1.*

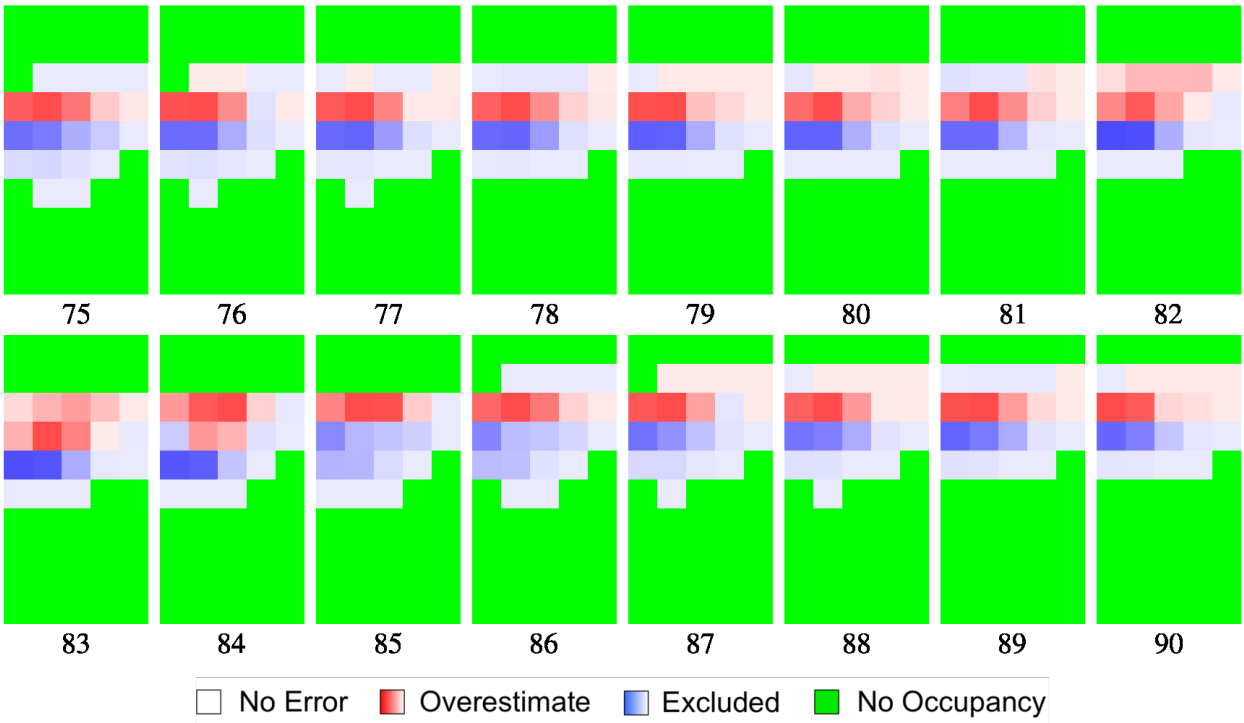

Figure 5.81: *Normalized excluded and overestimate heatmaps for graph "West4" for steps 75-90 and sample size of 2.*

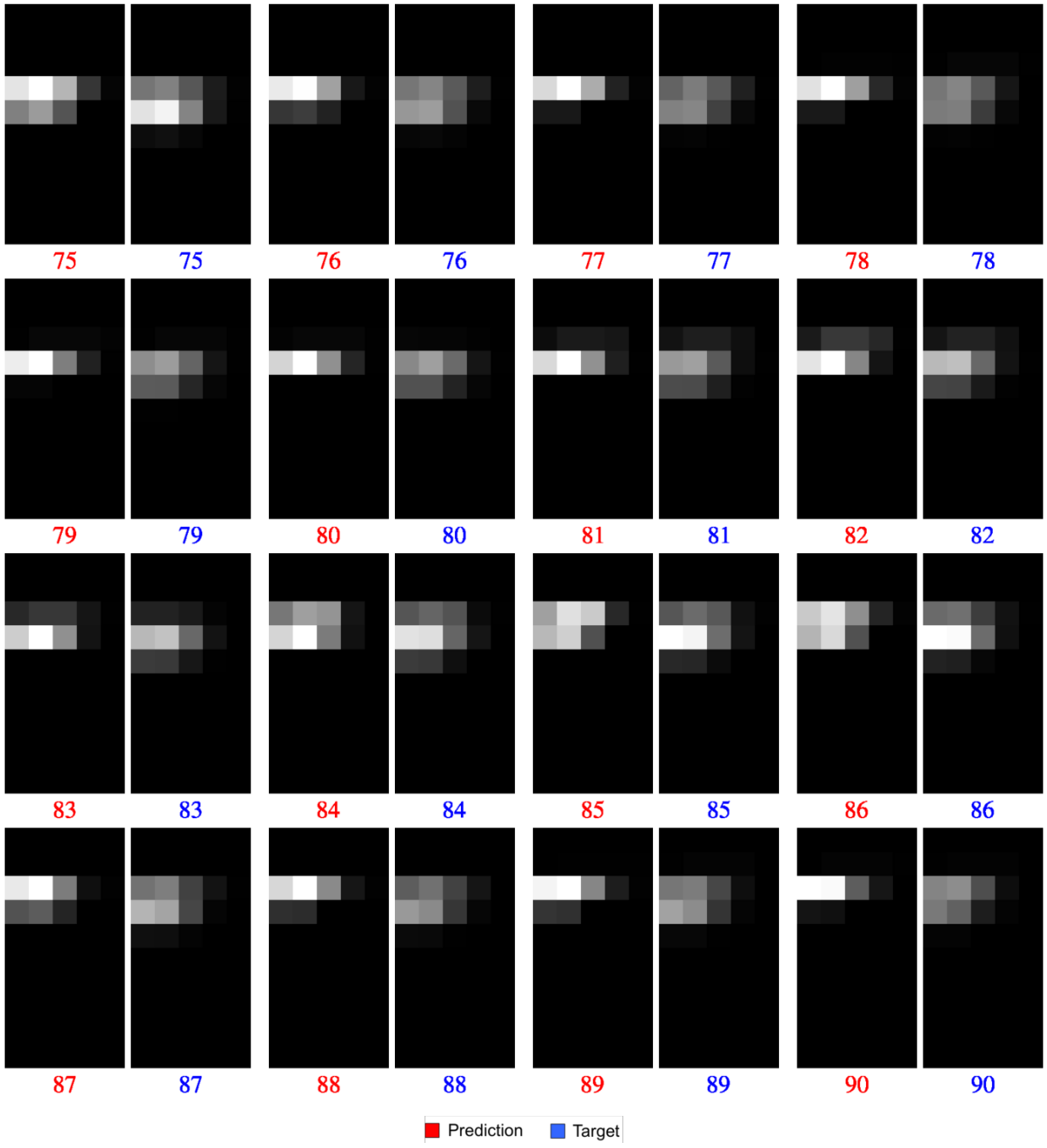

Figure 5.82: *Predicted occupancy for graph "West4" for steps 75-90 and sample size of 2.*

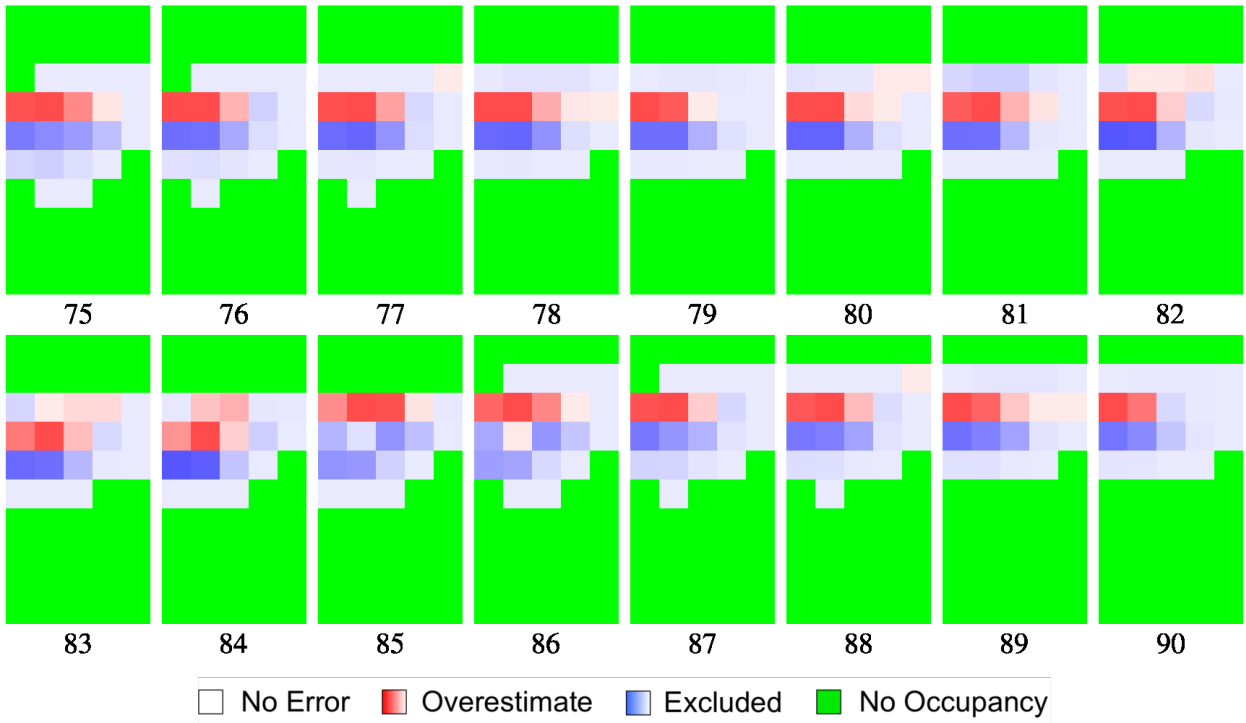

Figure 5.83: *Normalized excluded and overestimate heatmaps for graph "West4" for steps 75-90 and sample size of 4.*

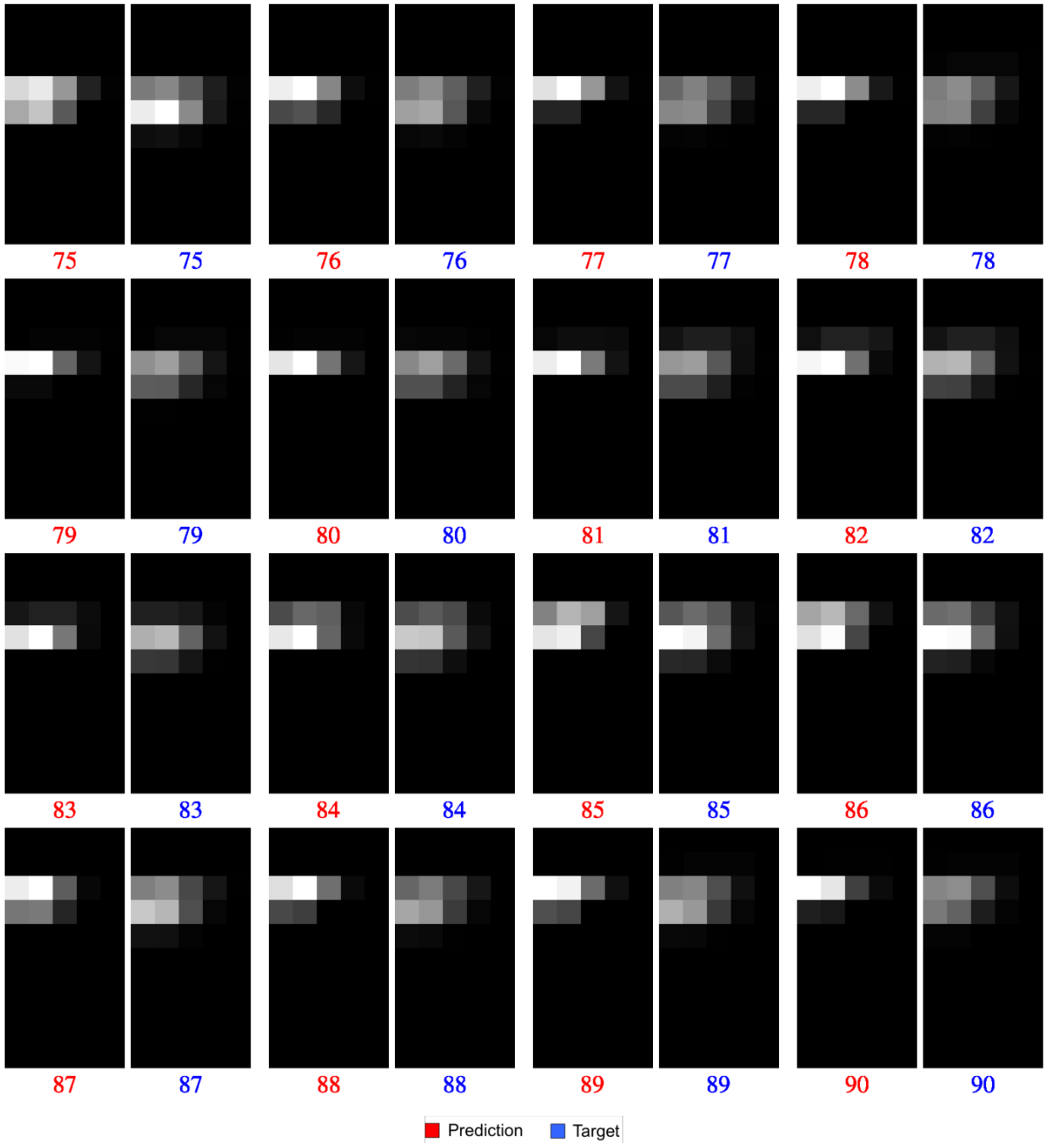

Figure 5.84: *Predicted occupancy for graph "West4" for steps 75-90 and sample size of 4.*

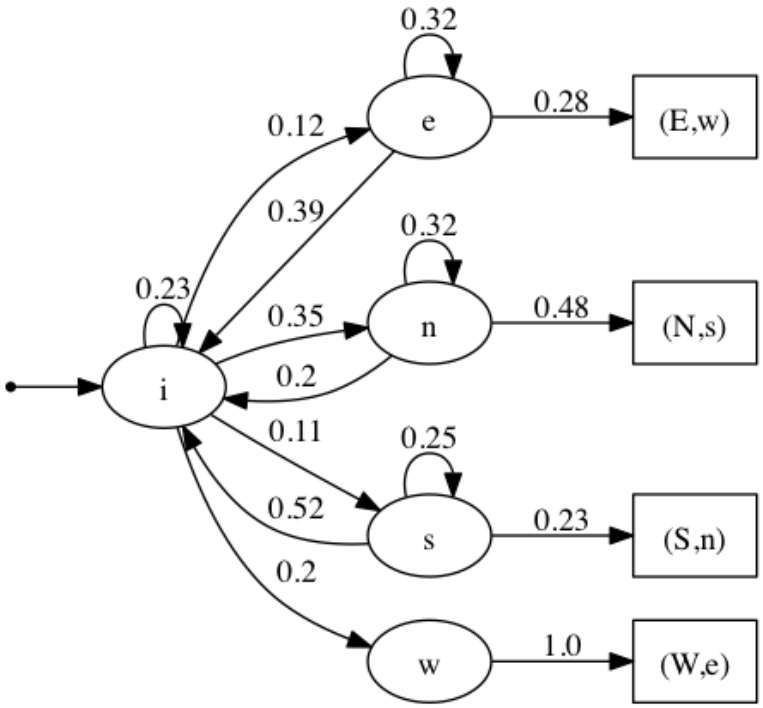

Figure 5.85: *Graph "West1 B" for batch execution "West1".*

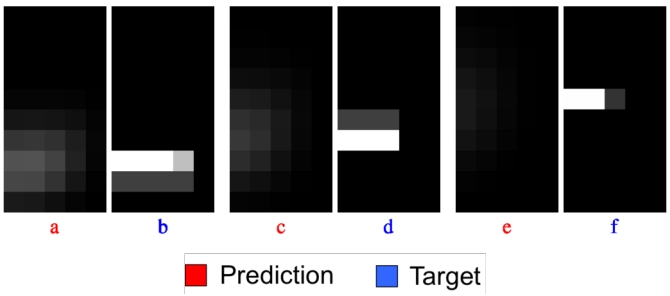

Figure 5.86: *Predicted occupancy for graph "West1 B" at steps 30, 50, and 70.*

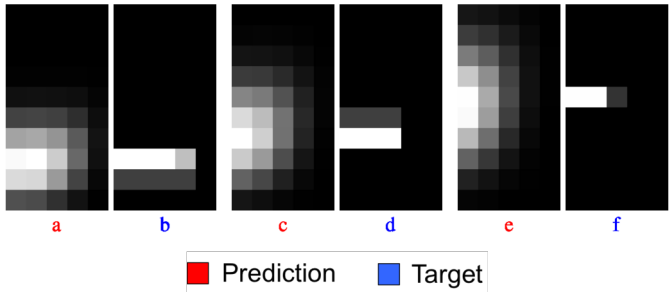

Figure 5.87: *Predicted occupancy for graph "West1 B" at steps 30, 50, and 70. Heat maps were adjusted individually.*

maps are adjusted individually so that, while we cannot use them to compare the occupancy between matrices, we can see where the highest occupancy is within each matrix. They show a prediction that correctly centers on the most important regions. In addition, it centers on the most important regions using only 5 real vertices (Figure 5.85).

On the other hand, the fact that it does not do a very good job of predicting the exact proportions of swarm in the regions is evident in the predictive value plot in Figure 5.90. However, rather than see this as a potential flaw in the prediction, we tend to see this as a limitation of the predictive value metrics. Upon inspection of the heat maps, it appears the prediction is actually good, depending the objectives. We leave finding one or more metric(s) that better capture the ability of smaller graphs to produce heat maps like this that give a more general idea of where the swarm is located to future work. Of course, even the existing metric may have some good things to say about the prediction. The fact that the discriminatory value remains above 25% suggests that even late in the batch execution the prediction is able to distinguish between those regions that are occupied and those that aren't, despite the fact that it fails to correctly predict the distributions in the regions. However, drawing that conclusion may require looking at the heat maps.

The excluded/overestimate heat maps in Figure 5.88 and occupancy heat maps in Figure 5.89 show heat maps for the local optima. They demonstrate that the prediction spreads

Figure 5.88: *Normalized excluded and overestimate heatmaps for graph "West1 B" for local optima.*

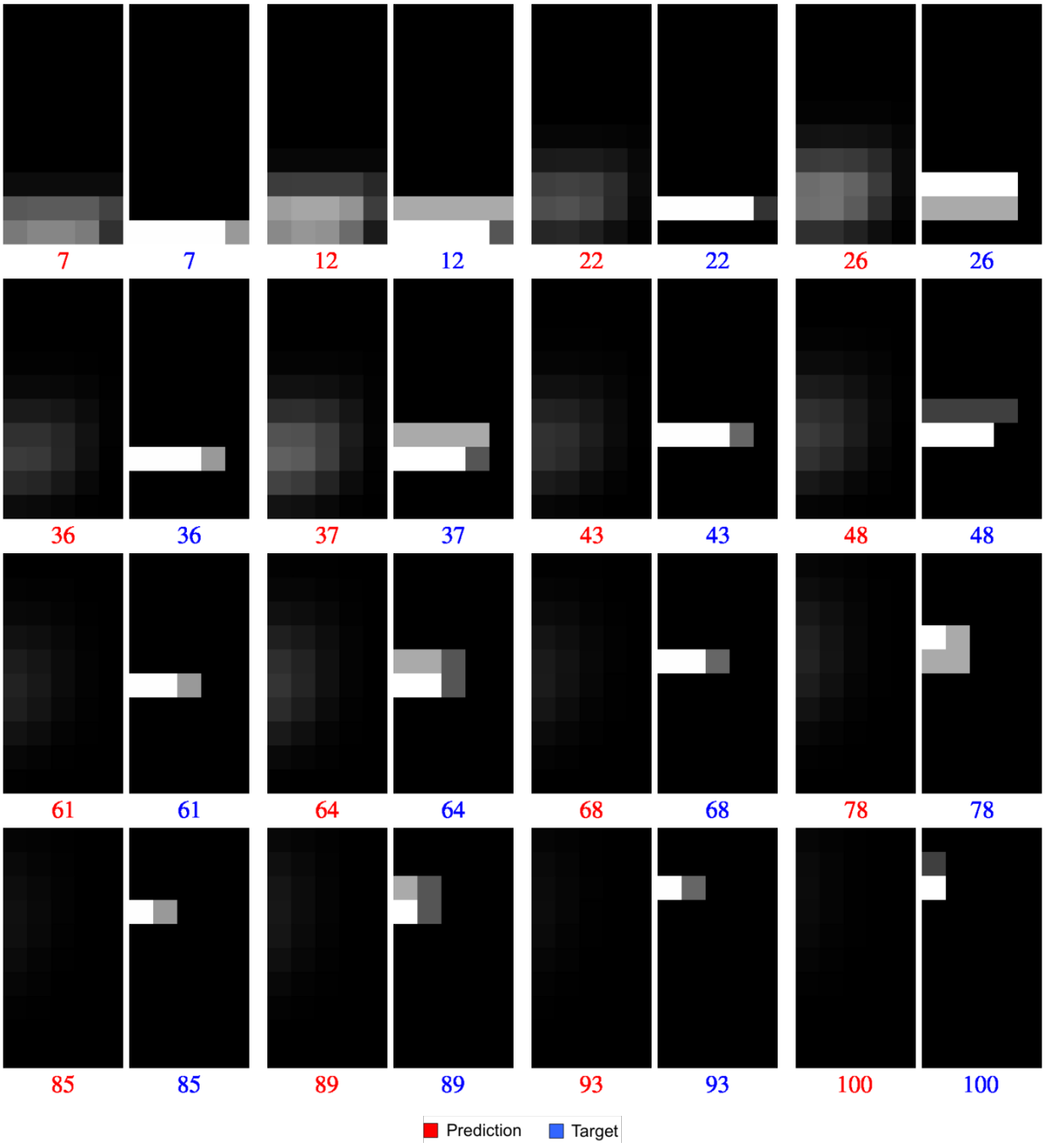

Figure 5.89: *Predicted occupancy for graph "West1 B" for local optima.*

very thin quickly (red) and that the oscillations in predictive value are due to the target occupying one row in some steps and two rows in others. The period associated with these oscillations alternates between 11 and 14. This may relate to the fact that the blue and green paths in the flow diagram for batch execution "West1" in Figure 5.41 are 11 and 12 steps long, respectively. Although, that doesn't explain the consistent alternation between a period of 11 and 14 over the course of the batch execution. It is possible that the graph contributes to this behavior. Answering this question is left to future work. The regional occupancy count in Figure C.31 shows how quickly the prediction swarm spreads out (cyan). A comparison with the predictive value supplement, also in Figure C.31, reveals that $included_s$ is higher and both $overestimate_s$ and $excluded_s$ lower when the target swarm occupies two rows. Hence, $inclusion_s$ and $discriminatory\_value_s$ are higher when the target swarm occupies two rows and lower when it occupies only one. In addition, $discriminatory\_value_s$ is generally higher than $inclusion_s$ beginning in step 4 because $excluded_s$ (yellow) is higher than $overestimate_s$ (magenta), which can be seen in the predictive value supplement plot in Figure C.31. The predictive value supplement plot in Figure C.31 also shows that $total\_occupancy > total\_predicted$ because the prediction swarm leaves the world at a higher rate than the target swarm. Furthermore, $overestimate_s = total\_predicted_s - inclusion_s$ (Equation 4.14) and $excluded_s = total\_occupancy_s - inclusion_s$ (Equation 4.13). It follows that $excluded_s > overestimate_s$ because $total\_occupancy_s > total\_predicted_s$.

5.12 Predicting Occupancy in Large Worlds

Once we have a probabilistic graph for a batch execution we can predict occupancy in batch executions that differ from it only in the dimensions of the world. Figure 5.92 shows how we are able to predict the target occupancy matrices of batch execution "West1 Large" using graph "West1 A" (Figure 5.13) that predicts batch execution "West1" in Section 5.2.

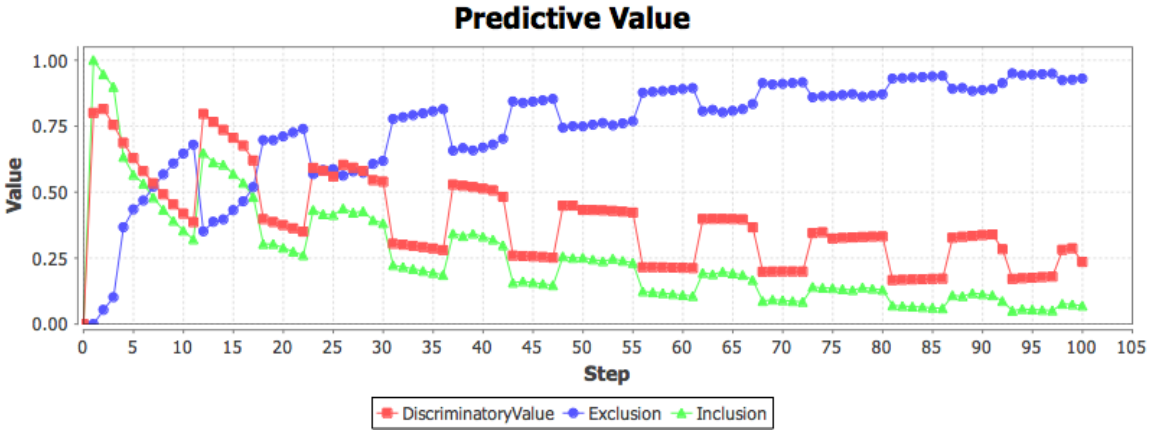

Figure 5.90: *Predictive value for graph "West1 B".*

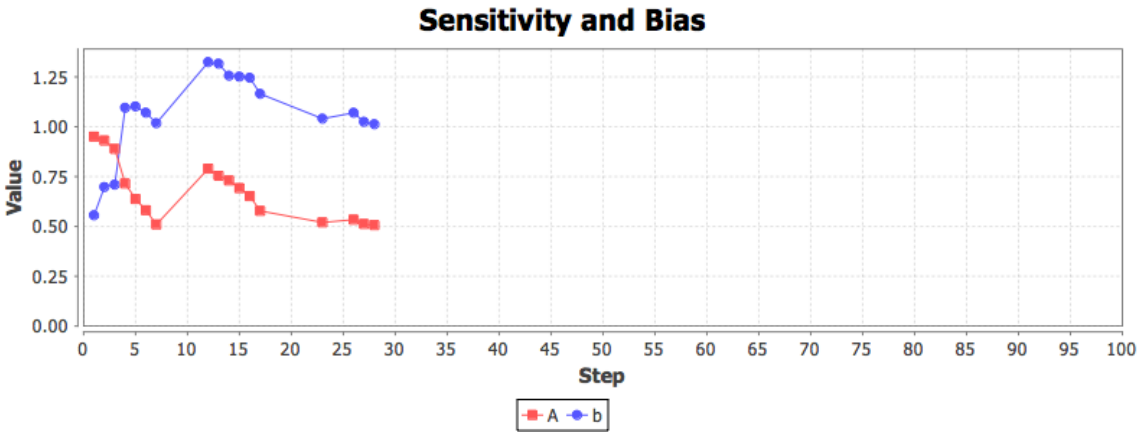

Figure 5.91: *Sensitivity and bias for graph "West1 B".*

The predictive value is given in Figure 5.93. The steps for this batch execution are shifted by 3, like they are for "West1". The predictive value for "West1 A" for batch execution "West1 Large" in Figure 5.93 looks noticably better than the predictive value for "West1" in Figure 5.14. To see why, see the excluded/overestimate heat maps for "West1 A" and "West1" in steps 89 through 100 in Figure 5.18. The predictive value for "West1 A" and "West1" decreases after step 89 due to low $included_s$ and high $overestimate_s$ because there is very little of the target swarm remaining the the world. What remains of the target swarm is in at most three regions after step 92 and most of it is in only one region. In contrast, Figure 5.96, reveals that due to the larger width of the world in batch execution "West1 Large", there is quite a bit more target swarm in the world. Figure 5.95 shows that there is quite a bit more regions with $excluded_s$ in the case of "West1 Large". As a result there remains quite a bit more $included_s$ in the large world (see the predictive value supplement for "West1 A" and "West1 Large" in Figure C.34) than in the small world (see predictive value supplement for "West1 A" and "West1" in Figure C.4) in the same steps due to the extra overlap between the prediction and target swarms. This leads to better $inclusion_s$ and $discriminatory\_value_s$ for the large world. The sensitivity and bias are shown in Figure 5.94. Like the predictive value, the sensitivity and bias for the later steps of "West1 Large" look like the sensitivity and bias for "West1" (Figure 5.15) in earlier steps with sensitivity above 0.75. The plot for b in Figure 5.94 shows a significant underestimate bias that appears to be increasing. That bias b results from a false-alarm rate that is high relative to the hit rate for those steps according to the isopleths in Figure 4.19. The hit rate and false-alarm rate can be seen in the sensitivity and bias supplement plot in Figure D.23.

Figure 5.97 shows the predicted occupancy for batch execution "West1 Large" using graph "West1 B" in Figure 5.85 that is used to predict batch execution "West1" in Section 5.11. Graph "West1 B" is the one learned by the genetic algorithm. The occupancy heat maps are adjusted individually to make the low prediction occupancy more visible.

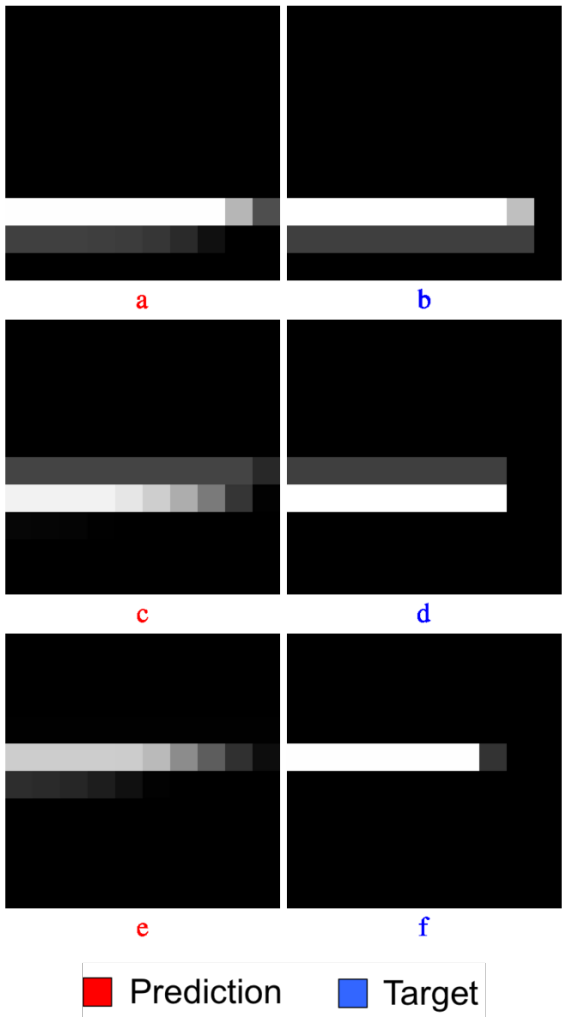

Figure 5.92: *Predicted occupancy for graph "West1 A" at steps 30, 50, and 70 in batch execution "West1 Large".*

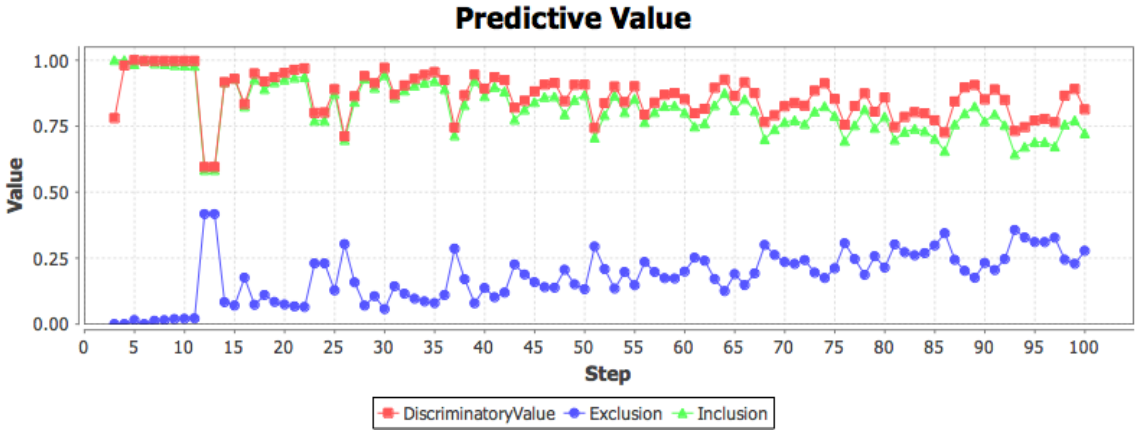

Figure 5.93: *Predictive value for graph "West1 A" in batch execution "West1 Large".*

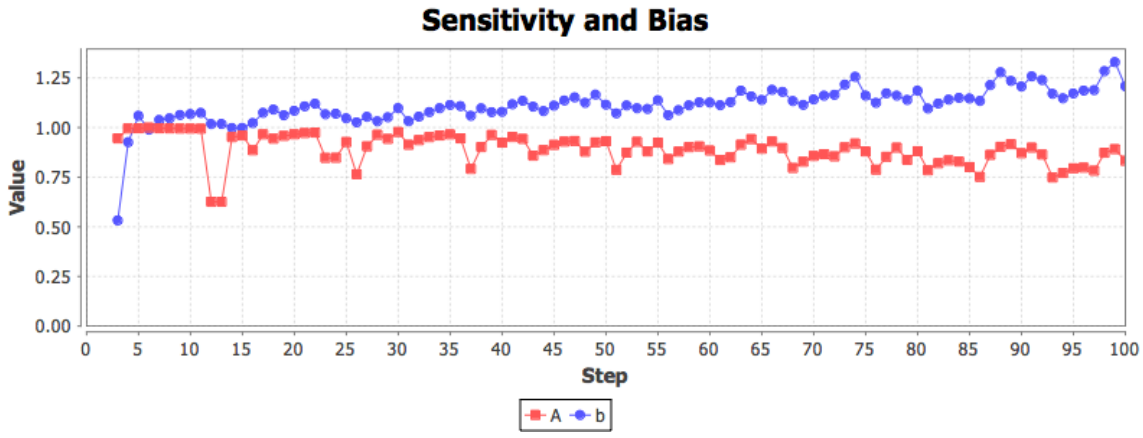

Figure 5.94: *Sensitivity and bias for graph "West1 A" in batch execution "West1 Large".*

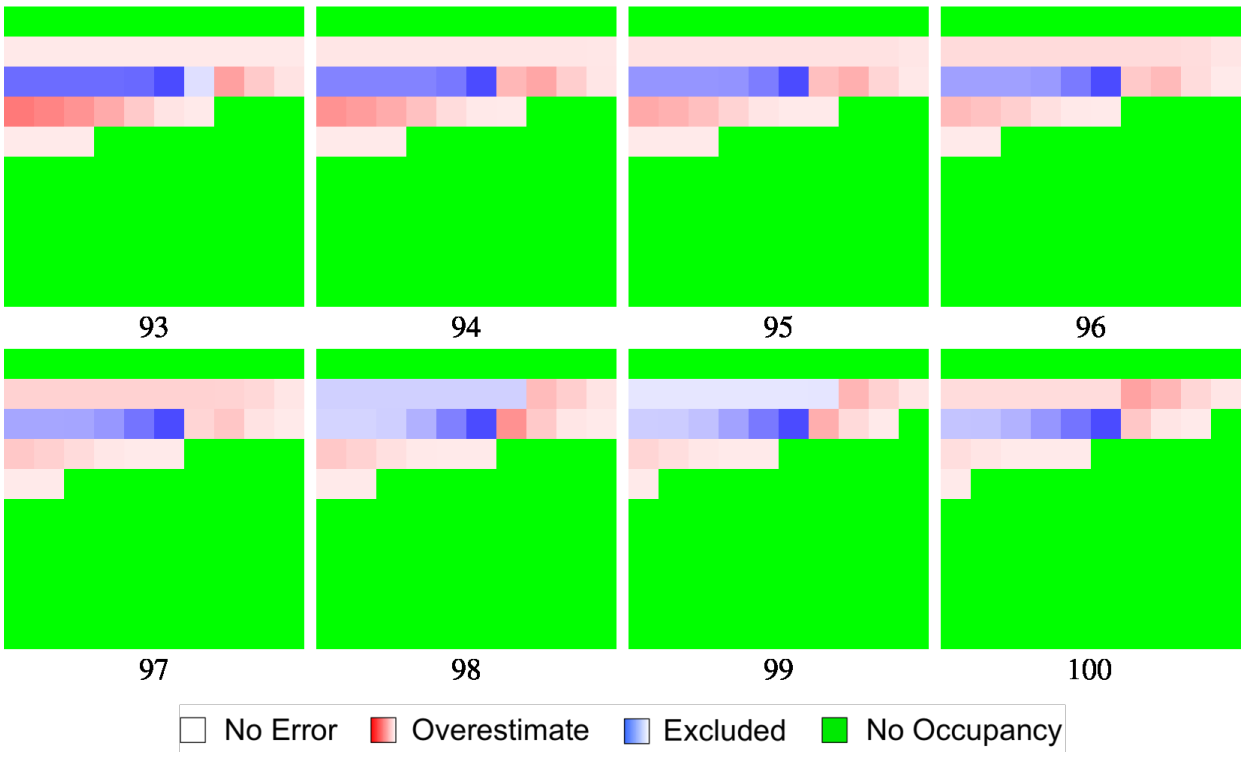

Figure 5.95: *Normalized excluded and overestimate heatmaps for graph "West1 A" for batch execution "West1 Large" for steps 93-100.*

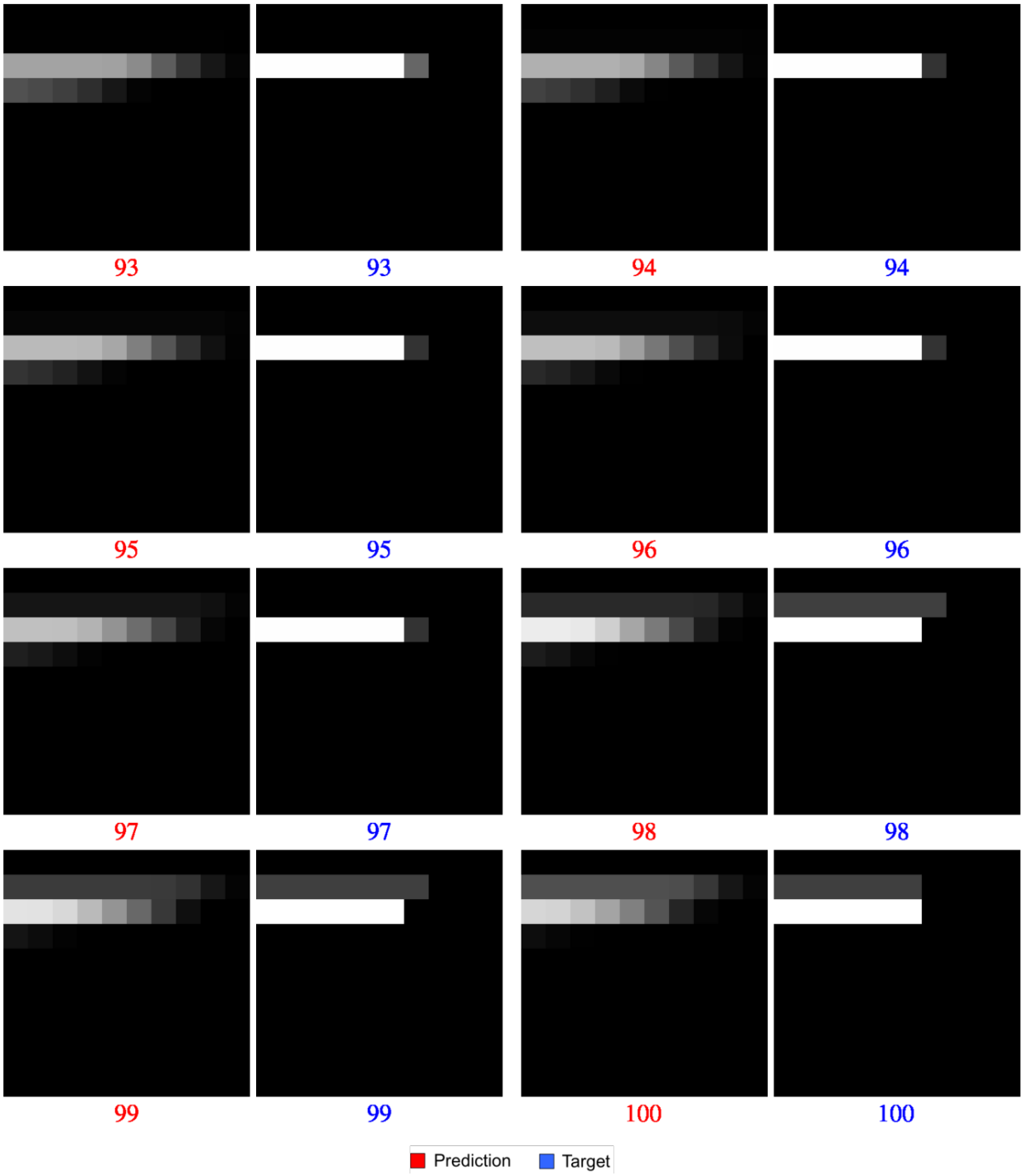

Figure 5.96: *Predicted occupancy for graph "West1 A" for batch execution "West1 Large" for steps 93-100.*

The predictive value is plotted in Figure 5.98. The predictive value plot shows that the predictive value for step 70 is in a trough, explaining why the prediction swarm looks a bit behind in the occupancy for step 70. Or rather, the predictive value for step 70 is in a trough because the prediction swarm is centered a row behind the target swarm. The occupancy in step 70 also looks slightly different from the occupancy in the same step for the smaller world in Figure 5.87. We attribute this to the fact that the swarm is so close to the edge of the world in the smaller world. The occupancy in the small world looks much more similar in step 30 and 50 when that is not so much the case. This was also the case for graph "West1 A" and batch execution "West1 Large". The period associated with the oscillations alternates between 11 and 14. This is similar to the alternating period for the smaller world (see page 163). As with the smaller world, it is not clear why the period alternates between 11 and 14. Further investigation is left to future work. The sensitivity and bias can be found in Figure 5.99.

The predicted occupancy for batch execution "West2 Large" using graph "West2 B" in Figure 5.34 that was used to predict batch execution "West2" in Section 5.5 is shown in Figure 5.100. The predicted value is in Figure 5.101. Figure 5.103 reveals why the oscillation is occuring. The batch execution is sometimes out of step. It is difficult to make out a pattern in the excluded and overestimate heat maps for steps 70 through 85 in Figure 5.104. However, the occupancy heat maps for steps 70 through 85 in Figure 5.107 suggest that the small increases in $inclusion_s$ and $discriminatory\_value_s$, like the ones in steps 76 and 77, come from small prediction swarm movements that barely change the shades in the prediction occupancy heat maps. Then the relatively large decreases in $inclusion_s$ and $discriminatory\_value_s$, like the ones in steps 75 and 78, result from target swarm movement that change the shades in the target occupancy matrices far more noticeably. The associated changes in e_p^s and o_p^s can be more easily seen in the mutually adjusted (meaning that the cells are divided by the largest value across all of the heat maps) excluded and overestimate

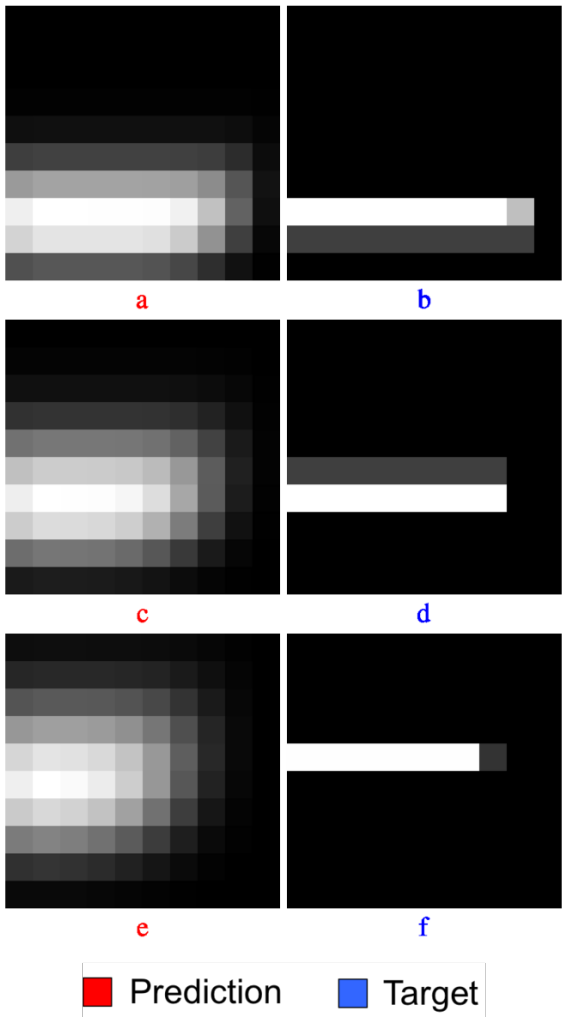

Figure 5.97: *Predicted occupancy for graph "West1 B" at steps 30, 50, and 70 in batch execution "West1 Large".*

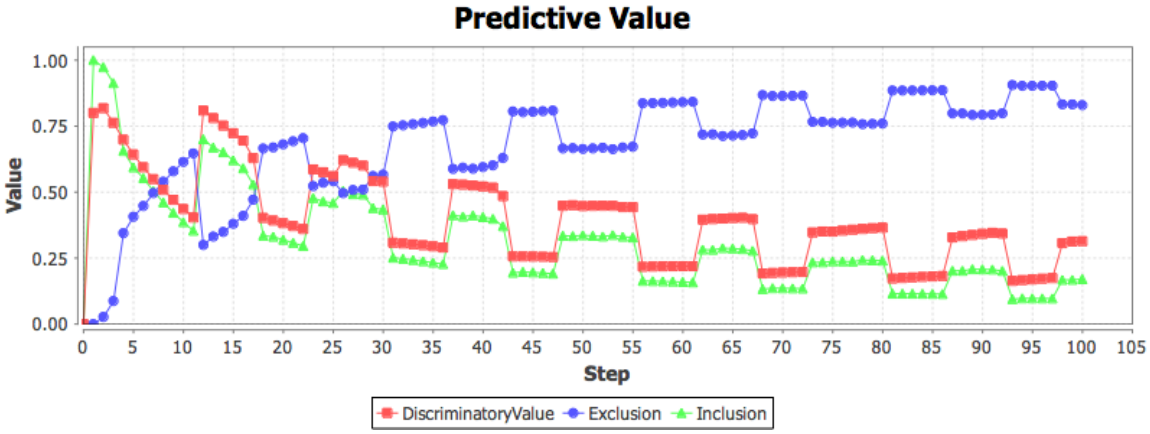

Figure 5.98: *Predictive value of graph "West1 B" for batch execution "West1 Large".*

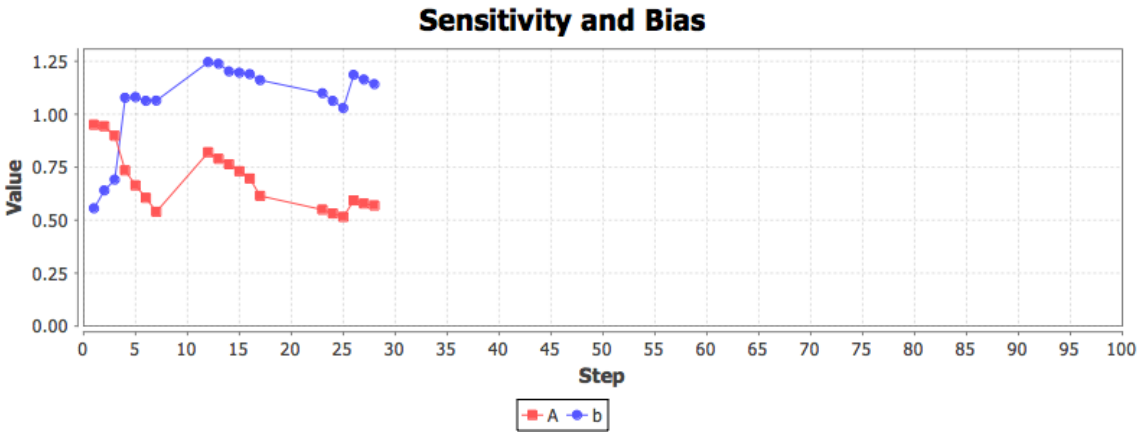

Figure 5.99: *Sensitivity and bias for graph "West1 B" for batch execution "West1 Large".*

heat maps in Figure 5.105. For example, the heatmaps for steps 75-77, 78-80, and 81-83 show the contribution to the incremental increases in predictive value in these steps (Figure 5.101). Then the heat maps in steps 75, 78, and 81 show a much larger contribution to the predictive value, since the colors are suddenly so much darker in these steps. Mutually adjusted included heat maps are shown in Figure 5.106. The period associated with the oscillations is 20. The period is the same for "West2 A" in the small world (see Figure 5.25). If the targets were obstacles, a robot could move through one of the regions within 15 steps. The length of 15 steps can be found by counting the steps along the longest diagonal path in Figure 4.2(c). The fact that the period is longer than 15 suggests that there may be a contribution from the targets. The robots may repeat tagging moves in each row since the batch execution is deterministic. The prediction may also contribute to the period. Further investigation of the periodicity is left for future work. The sensitivity and bias can be found in Figure 5.102.

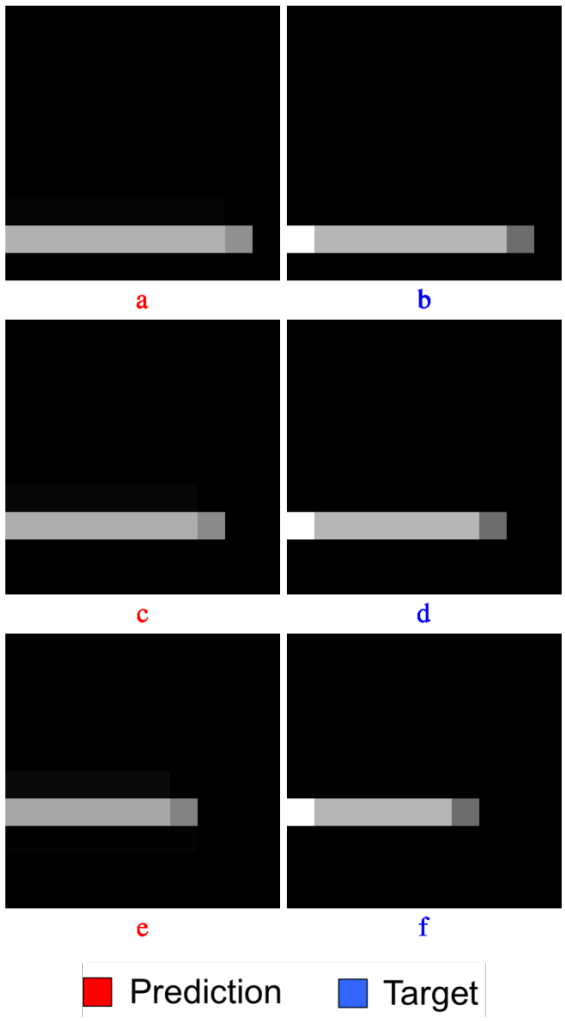

Figure 5.100: *Predicted occupancy for graph "West2 B" at steps 30, 50, and 70 for batch execution "West2 Large".*

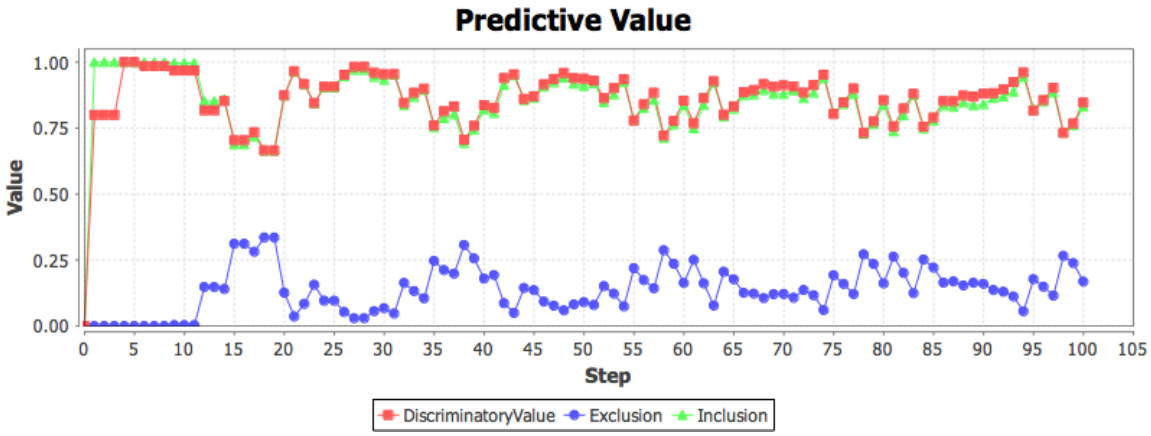

Figure 5.101: *Predictive value of graph "West2 B" for batch execution "West2 Large".*

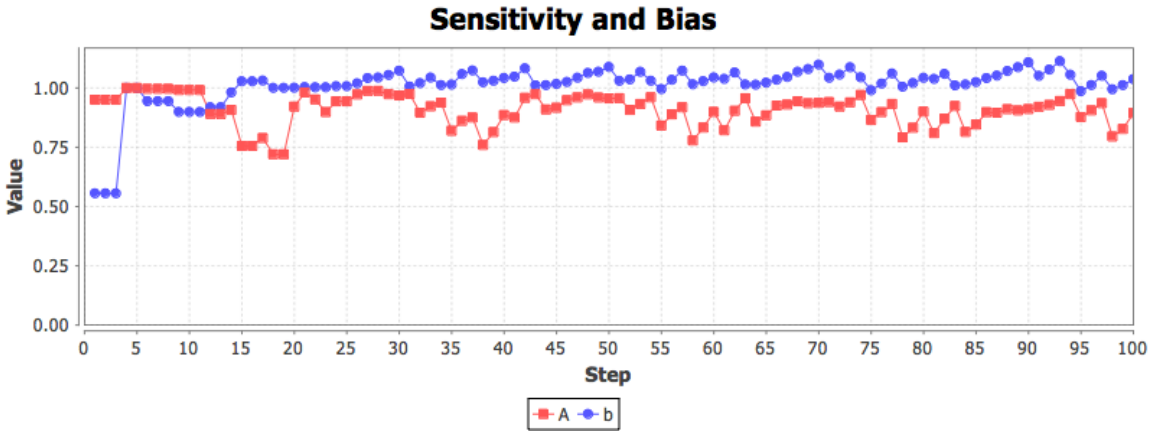

Figure 5.102: *Sensitivity and bias for graph "West2 B" for batch execution "West2 Large".*

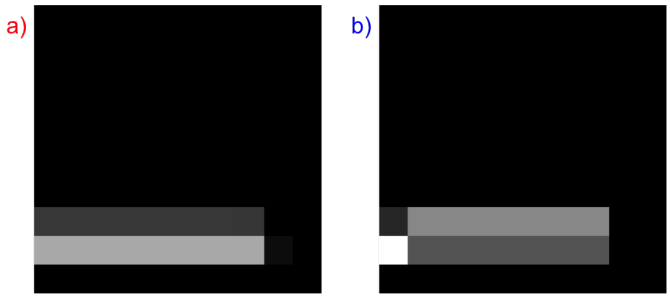

Figure 5.103: *Predicted occupancy of graph "West2B" at step 38 for batch execution "West2 Large".*

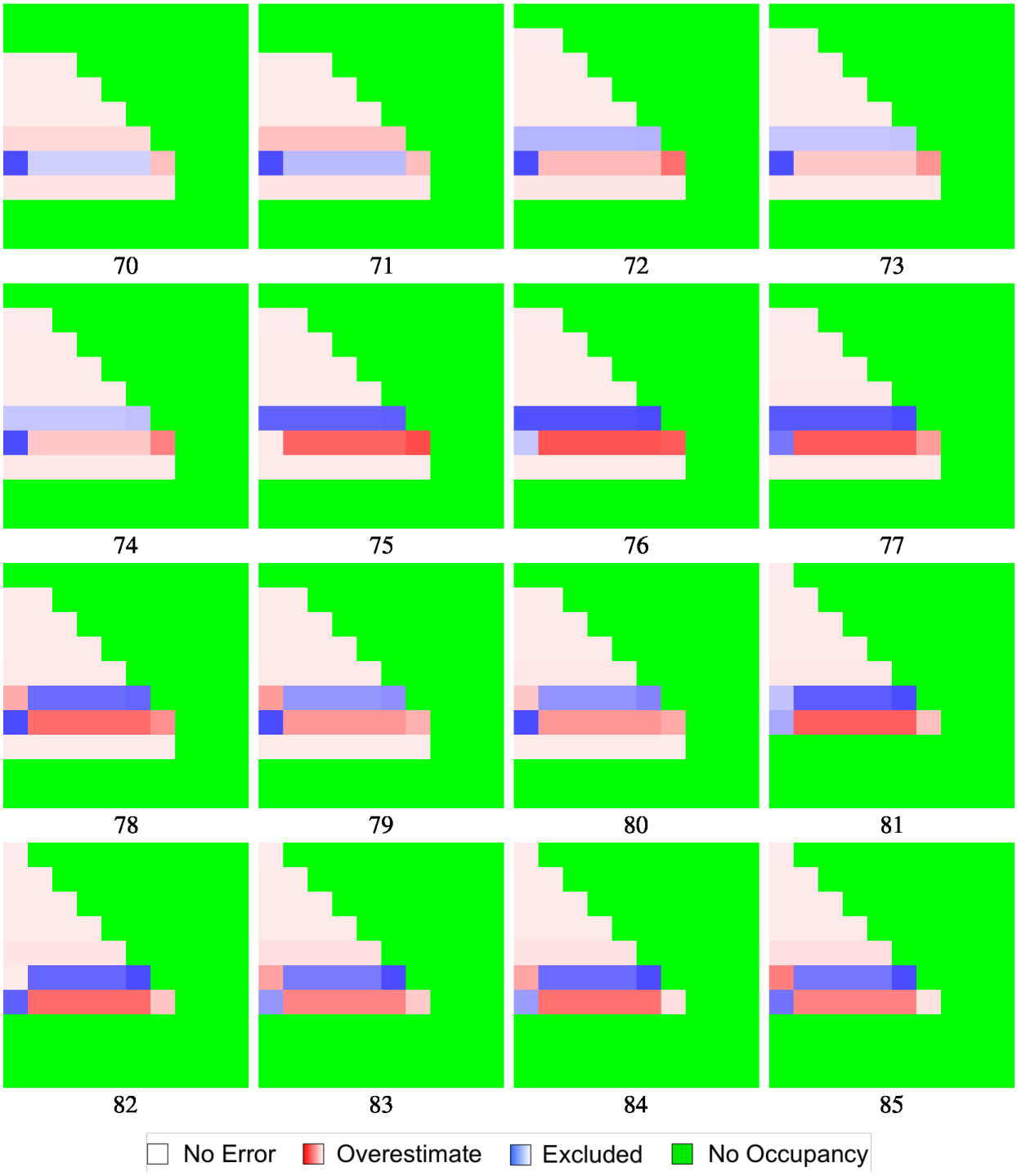

Figure 5.104: *Normalized excluded and overestimate heatmaps for graph "West2 B" and "West2 Large" for steps 70-85.*

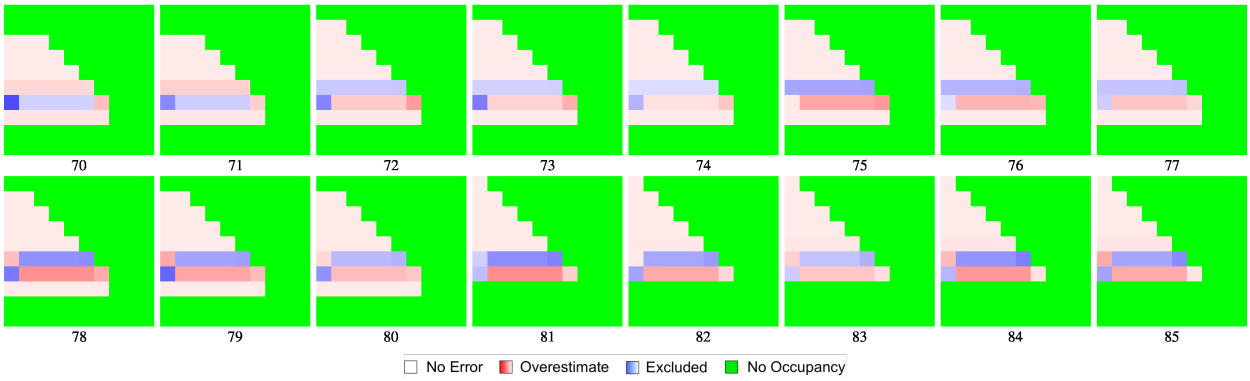

Figure 5.105: *Mutually normalized excluded and overestimate heatmaps for graph "West2 B" and "West2 Large" for steps 70-85.*

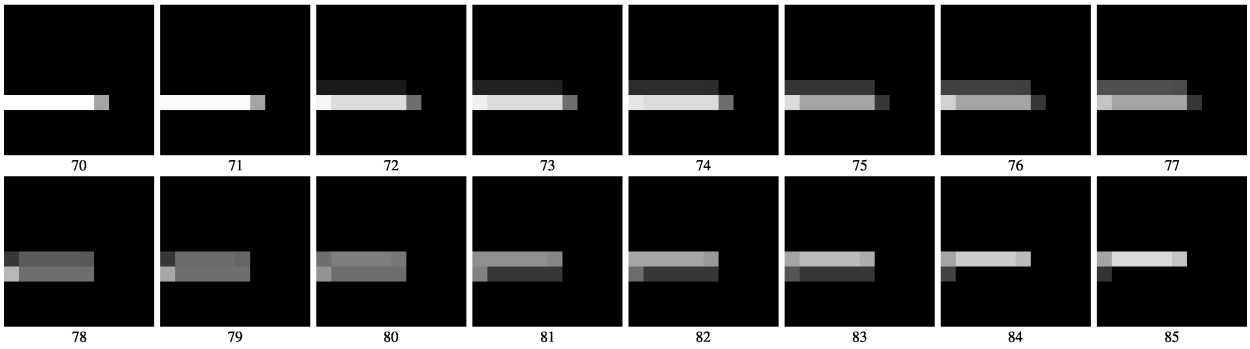

Figure 5.106: *Mutually normalized included heat maps for graph "West2 B" and "West2 Large" for steps 70-85.*

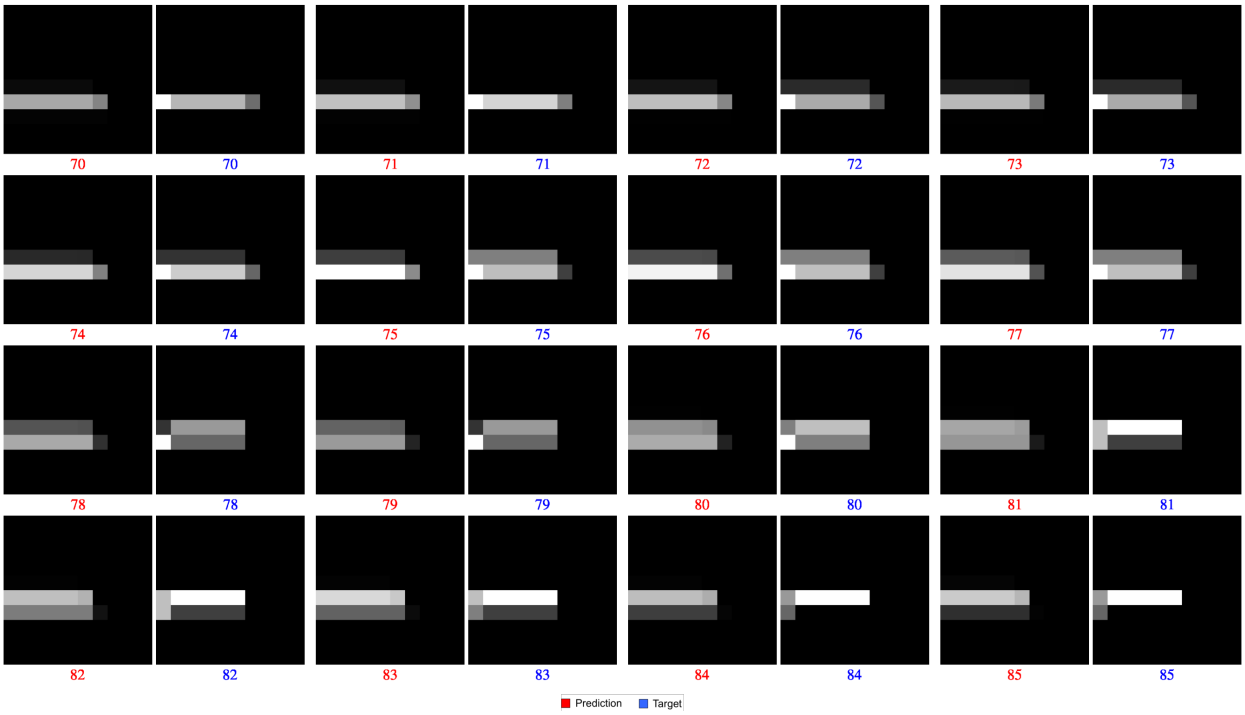

Figure 5.107: *Predicted occupancy for graph "West2 B" and "West2 Large" for steps 70-85.*

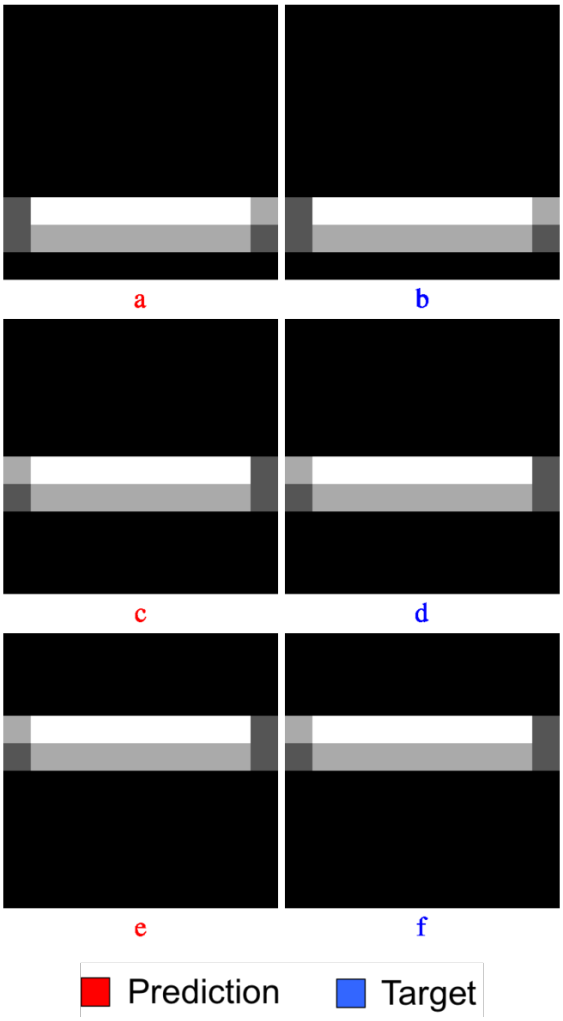

Figure 5.108: *Predicted occupancy of graph "West3 C" at steps 30, 50, and 70 batch execution set "West3 Large".*

The predicted occupancy for batch execution "West3 Large" using graph "West3 C" in Figure 5.63 that was used to predict batch execution "West3" in Section 5.8 is shown in Figure 5.108. The predictive value is plotted in Figure 5.109. Graph "West3 C (I)" in Figure 5.67 is used for the first row like it was in Section 5.9. The sensitivity and bias can be found in Figure 5.110.

The predicted occupancy for batch execution "West4 Large" using graph "West4" (Section 5.10) is shown in Figure 5.111. The predictive value can be found in Figure 5.112.

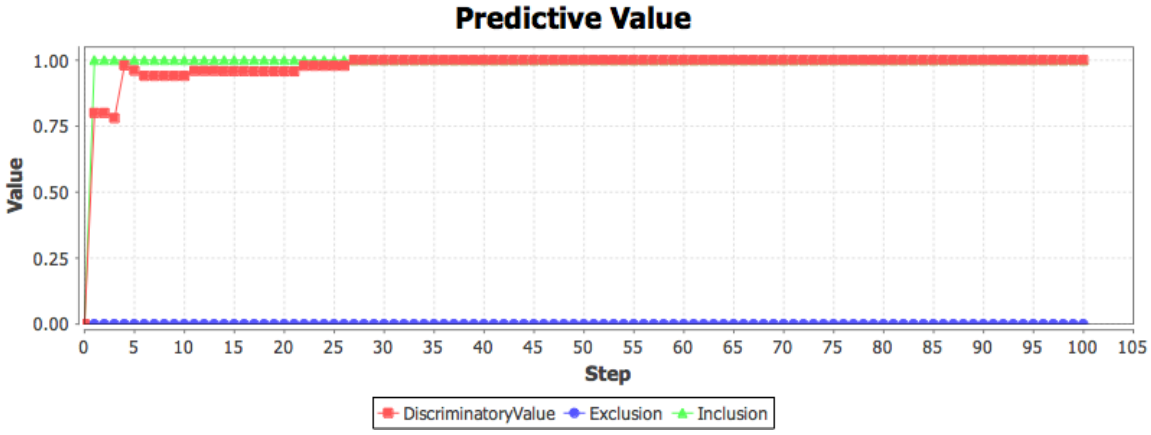

Figure 5.109: *Predictive value of graph "West3 C" for batch execution "West3 Large".*

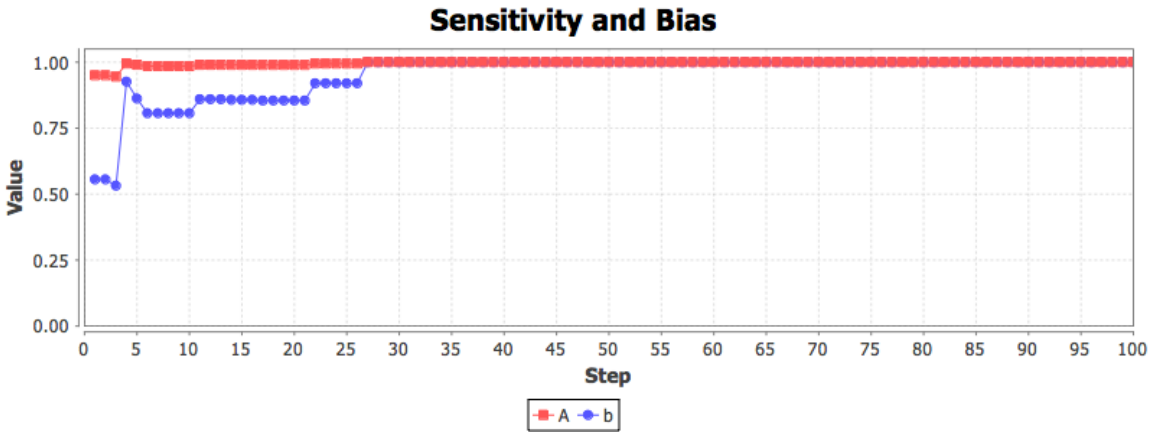

Figure 5.110: *Sensitivity and bias for graph "West3 C" for batch execution "West3 Large".*

Graph "West4 (I)" is used for the first row. The steps are shifted by 1 just like they are for batch execution "West4".

Table 5.8: *Periods by prediction.*

Prediction	Period
East	none
West1 A	none
West2 A	20
West2 B	none
West3 A	11
West3 B	11
West3 C	none
West4	11
West1 B	11/14 (alternating)
West1 A Large	none
West1 B Large	11/14 (alternating)
West2 B Large	20
West3 C Large	none
West4 Large	11

5.13 Periodicity in Predictive Value

The periodicity in the predictive value of each prediction is shown in Table 5.8. Some predictions do not show obvious periodicity. These predictions are "East" (Figure 5.4), "West1 A" (Figure 5.14), "West2 B" (Figure 5.36), "West3 C" (Figure 5.65), and "West1 A" against "West1 Large" (Figure 5.93). Four predictions show periodicity that is close to the length of paths that the target swarm follows through the square pattern. The predictive value for the predictions "West3 A" (Figure 5.44), "West3 B" (Figure 5.53), "West4" (Figure 5.72), and "West4" against "West4 Large" (Figure 5.112) fall into this category. There are also predictions where the lengths of the paths through the square pattern do not seem to be enough to explain the period. These include the predictive value for the graphs "West2A" (Figure 5.25) and "West2B" against "West2 Large" (Figure 5.101) where repeating target tagging behaviors in each row may make a contribution. They also

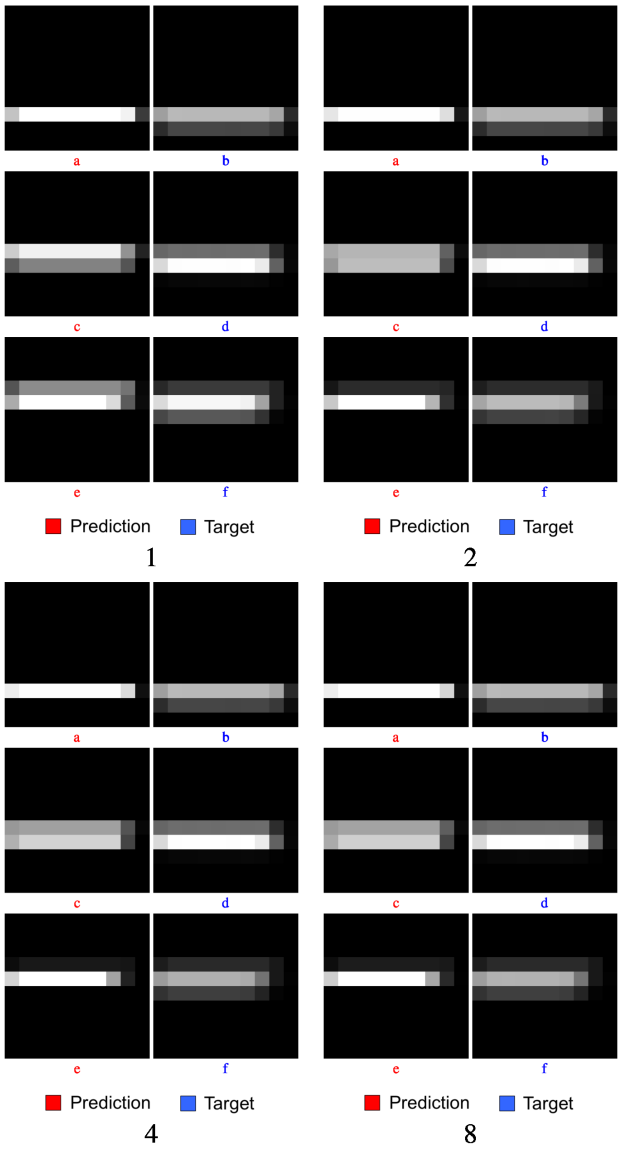

Figure 5.111: *Predicted occupancy of graph "West4" at steps 30, 50, and 70 for batch execution "West4 Large".*

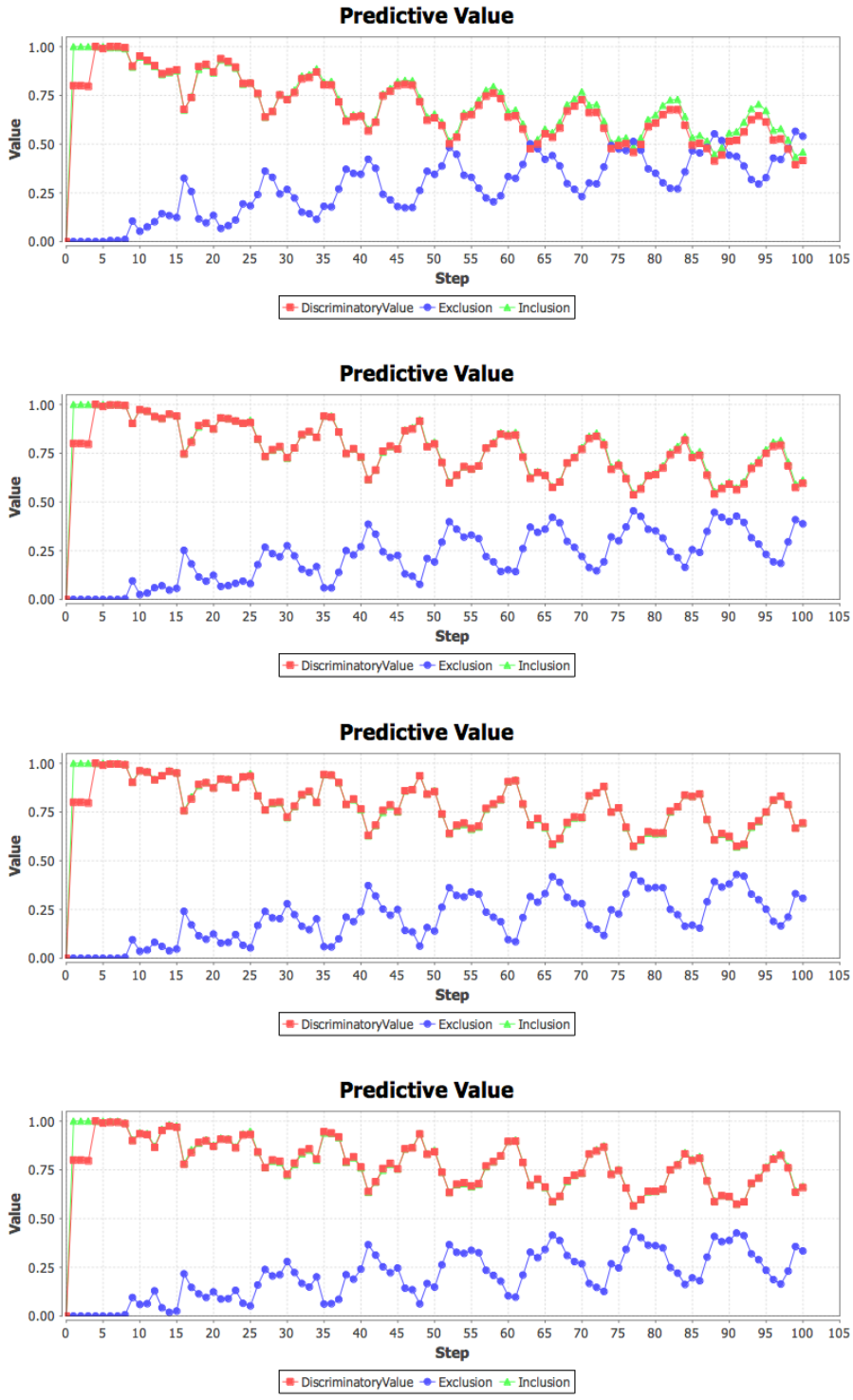

Figure 5.112: *Predictive value of graph "West4" for batch execution "West4 Large".*

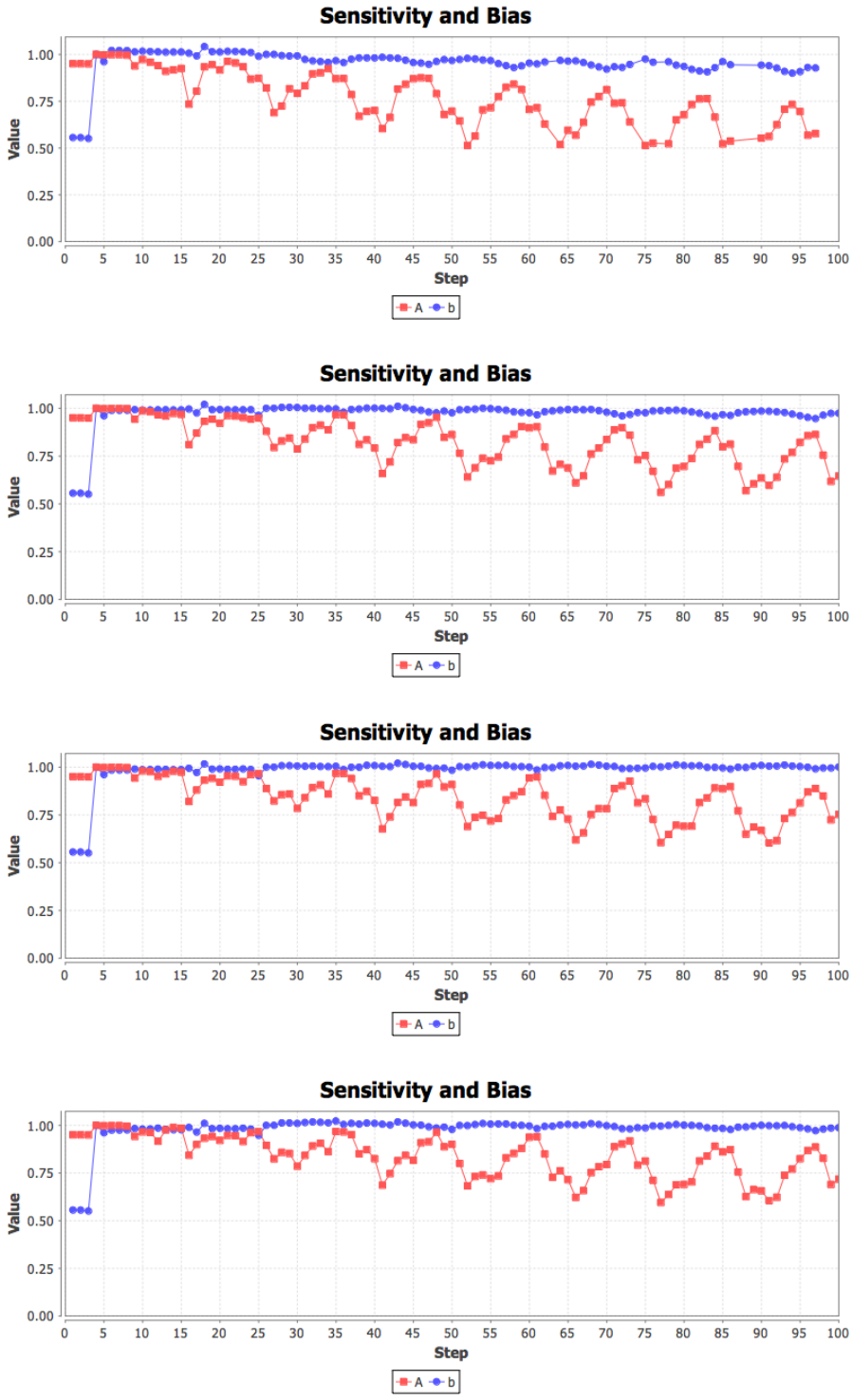

Figure 5.113: *Sensitivity and bias for graph "West4" for batch execution "West4 Large".*

include the predictions "West1 B" (Figure 5.90) and "West1 B" against "West1 Large" (Figure 5.98) where the period alternates between 11 and 14. The world for batch execution "West1" does not contain targets and the primary paths that the target swarm follows through the world include lengths 9, 11, and 12. But these lengths do not seem to help explain a period of 14 or why the period alternates between 11 and 14. One explanation is that the prediction swarm makes a contribution. Another explanation is that robot collisions make a contribution. It could also be a combination of the two. A more thorough investigation into the periodicity is left to future work.

Chapter 6

Discussion

We presented several probabilistic graphs in Chapter 5 that captured the direction and rate of movement of swarms in world executions. We used the inclusion and discriminatory value metrics to objectively assign the graphs a measure of quality based on their ability to track the majority of the swarm with the majority of the predicted occupancy.

6.1 On the Predictive Value Metrics

The predictive value metrics were objective and quantified the value of predicted occupancy in each region. They gave us a way to determine how spread out the prediction was or how it was able to distiguish between occupied and unoccupied regions, rather than simply containing them. But while this metric may continue to be useful for the purpose of quantatively assigning a measure of quality to predictions that are meant to track the proportion of the swarm in each region, it seems that it may be overly restrictive for some interesting and useful predictions.

There are predictions that, while failing to predict the exact occupancy in the regions, do a very good job of centering on and surrounding the most heavily occupied regions.

The associated heat maps are still very informative. Take for example the individually adjusted heat maps of "West3 A" in Figure 5.46 and "West1 B" in Figure 5.87. They correctly matched the direction and rate of the swarms in the associated target occupancy matrices, even if the information about the specific regions was lost. Graph "West1 B" in Figure 5.85, was able to predict the direction and rate with under half of the real vertices of its counterpart, "West1 A" in Figure 5.13, even though "West1 A" resulted in better predictive value. In fact, graph "West1 A", while obtaining better predictive value than "West1 B" and better occupancy (Figure 5.12), was also an approximation. We could have produced a perfect prediction if we had followed the steps that produced graph "West3 C" in Figure 5.63. The occupancy for "West3 C" is shown in Figure 5.64. Obviously, graph "West3 C" had perfect predictive value (Figure 5.65). It is a method that the flow in Figure 5.41 suggests would have worked similarly well for batch execution "West1". And there is "West2 B" in Figure 5.34, where despite not having a repetitious square pattern to work with due to the square pattern and target interference, we were able achieve very high predictive value and nice heat map predictions as well.

These predictions, while not attempting to track the exact movements of individual robots, were still able to pin down the exact regions containing the majority of the swarm. Unless we are concerned with the exact amount of the swarm that occupies each region at every step, it may be enough to know that our prediction is very close or that it centers on and surrounds the swarm. Depending on the application, we might not even care if the prediction is not centered on the swarm, if for example, it is dense and directly adjacent to the swarm, or perhaps just ahead of it. These different variations in heat map types and interpretations can provide even more flexibility to use these graphs to predict the locations of swarms in more challenging environments by providing additional modeling and prediction methods. If the heat maps for the predictions with oscillations remain useful, a metric that assigns a high value for containing the the swarm in a broader sense will

help with oscillations that are not altogether undesirable. This may become particularly important with more complicated square patterns and non-deterministic batch executions. For some of these batch executions, it may be more difficult or perhaps even impossible to find local patterns of behavior to predict the swarm locations exactly as we have in this work. In more open, uncertain, and continuous environments, we might be interested in the general location of the swarm if we are unable to determine how much of it is in exact regions at specific times. So a metric that provides higher values to predictions that center on and tightly surround the swarm may be beneficial.

6.2 Oscillations in Predictive Value

Work needs to be done to better understand the oscillations and associated periodicity that often show up in the predictive value of some predictions. This may involve looking closer into the periodicity. It may be possible to quantify it. Quantifying the periodicity may reveal patterns and allow comparison. We may also look more into shifting and its effects on periods. For example, for a given graph, we could focus on one period in the predictive value and vary the phase for the graph by shifting the steps and note how the predictive value changes. This could be useful since much of the oscillations seem to be caused by the prediction and target occupancy being out of phase. An interesting question might be how we can use information about a period to improve a graph. It may also be enough to have a closer look at the underlying world executions. There may be something very straightforward going on that contributes to the more interesting periods, such as the period of 20 for graph "West2 A" (Figure 5.25) where a repeat target tagging behavior may be involved. We may try tuning edge probabilities in the graphs in cases where the prediction swarm makes a significant contribution to the predictive value. This technique will be less useful for graphs like "West3 A" (Figure 5.44) where the prediction swarm is so spread out as to not make a

significant contribution to the predictive value. Cases where the period alternates between two values are especially interesting, such as for graph "West1 B" (Figure 5.85) where the period alternates between 11 and 14 (see Figure 5.90). Understanding the oscillations may also involve sending only one robot into the batch executions in an attempt to account for the contribution of from collisions. The contribution from targets could be investigated too. The period in the predictive value for graph "West2 A" (Figure 5.24) in Figure 5.25 is 20 although the robots could move through the world in 15 steps and this could be attributed, in part, to the targets.

6.3 Creating Graphs

A more flexible metric and a better understanding of the initial conditions will help in the creation of new methods of creating the graphs. We already have a few tools and tricks in hand for their design, but knowing more about how they can be used, quantified, and how to automate their construction will help. Continuing with the evolutionary approach is one possible direction. We already have a genetic algorithm implementation for evolving the graphs, but it will take more time to refine the heavily parameterized process. Hill climbing might be a better alternative, particularly for smaller graphs that are only meant to center on and surround the swarm. The graphs don't seem to be a very complex representation and a hill climbing algorithm will have fewer parameters. But the manual creation of the graphs may continue to be an important method. And the tools that we use to analyze the graphs when creating them by hand might inform other, similarly simple representations (most likely variations of the existing graphs) for predicting swarm location.

6.4 Practical Applications

Next, we need to test the graphs with physical swarms. Realistic simulations of real robots might be a next step in that direction. It is important to show that the graphs are useful in practice. But it will also provide us with experience in practical considerations that might make it into our more abstract simulations, reducing the gap between the simulations and use in physical robots.

Finally, there are a number of possible future directions. It might be useful to mix square patterns in batch executions, as that would increase the applicability of the approach since we would be able to use it for quite a few more worlds than we can currently with worlds of a single square pattern. We could try using larger square patterns or even nesting square patterns, and the associated graphs, to make predictions about swarms in larger worlds. This might involve replacing the graph of the subpatterns with a single graph for the meta-pattern, perhaps at multiple levels. Creating parameterized graphs is another possibility. For example, it might be possible to model the "West1", "West3", and "West4" batch executions and many variations in-between by parameterizing the edges of one vertex in graph "West3 C" based on the desired right turn probability. Then we could predict occupancy in the aforementioned batch executions with one graph by using a right turn probability of 0.0, 1.0, or 0.5, respectively. It would allow for testing others in between as well. Note that this would be an alternative to many world executions, since the world executions with a right turn probability between 0.0 and 1.0 are non-deterministic. Parameterized graphs would give us even more flexibility and power to play with new world size and parameter value combinations before simulation or deployment. Down the road it may even be possible to use the graphs to play with new square patterns and parameter values before implementing them in simulation. Lastly, we may not be able to count on static worlds in many real world applications such as search and rescue. One possible direction would be to deploy scouts

and study their behavior to produce a probabilistic graph. Then predictions about a swarm can be made with this new graph, perhaps within minutes of an event such as a disaster.

Chapter 7

Conclusion

It is encouraging that the complex behavior produced by different combinations of square patterns and parameter values can be represented using simple probabilistic graphs. While these combinations can result in very complex behavior, graphs are able to anticipate the result. The predictions reduce the actions of 25 to 50 robots to a single matrix multiplication. These events may include collisions with other robots, obstacles, or target interference and we are able to describe the behavior collectively using simple graphs that capture the behavior at the mean. It is also encouraging that once produced, a graph can be used to make predictions about ever larger worlds.

For now the approach is confined to discrete simulations, but with the proper setup, it could be tested on real robots. If tested successfully on real robots, the approach could be used to predict the location of swarms in highly patterned environments such as country roads, fields, and city blocks. It may also inform methods that are less dependent on a single square pattern, such as the use of multiple square patterns or nested square patterns, making the approach applicable in more environments.

Bibliography

Abukhalil, T., M. Patil, and T. Sobh (2013). Survey on decentralized modular swarm robots and control interfaces. *International Journal of Engineering (IJE) 7*(2), 44.

Agassounon, W. (2003). *Modeling artificial, mobile swarm systems*. Ph. D. thesis, California Institute of Technology.

Agassounon, W. and A. Martinoli (2002). A macroscopic model of an aggregation experiment using embodied agents in groups of time-varying sizes. In *Proc. of the IEEE Conf. on System, Man and Cybernetics (SMC), Hammamet, Tunisia*, pp. 250–255.

Agassounon, W., A. Martinoli, and K. Easton (2004). Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes. *Autonomous Robots 17*(2-3), 163–192.

Anderson, C., G. Theraulaz, and J.-L. Deneubourg (2002). Self-assemblages in insect societies. *Insectes sociaux 49*(2), 99–110.

Aznar, F., M. Sempere, M. Pujol, R. Rizo, and M. Pujol (2014). Modelling oil-spill detection with swarm drones. In *Abstract and Applied Analysis*, Volume 2014. Hindawi Publishing Corporation.

Barca, J. C. and Y. A. Sekercioglu (2013). Swarm robotics reviewed. *Robotica 31*(3), 345–359.

Bayindir, L. and E. Şahin (2007). A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences 15*(2).

Bayindir, L. and E. Şahin (2009). Modeling self-organized aggregation in swarm robotic systems. In *Swarm Intelligence Symposium, 2009. SIS'09. IEEE*, pp. 88–95. IEEE.

Beni, G. (2005). From swarm intelligence to swarm robotics. In *Swarm Robotics*, pp. 1–9. Springer.

Beni, G. and J. Wang (1993). Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, pp. 703–712. Springer.

Berman, S., A. Halász, M. A. Hsieh, and V. Kumar (2008). Navigation-based optimization of stochastic strategies for allocating a robot swarm among multiple sites. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 4376–4381. IEEE.

Berman, S., A. Halász, V. Kumar, and S. Pratt (2006). Algorithms for the analysis and synthesis of a bio-inspired swarm robotic system. *4433*, 56–70.

Berman, S., Á. Halász, V. Kumar, and S. Pratt (2007). Algorithms for the analysis and synthesis of a bio-inspired swarm robotic system. In *Swarm Robotics*, pp. 56–70. Springer.

Berman, S., Q. Lindsey, M. S. Sakar, V. Kumar, and S. C. Pratt (2011). Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proceedings of the IEEE 99*(9), 1470–1481.

Berman, S. M. (2010). *Abstractions, analysis techniques, and synthesis of scalable control strategies for robot swarms*. Ph. D. thesis, UNIVERSITY OF PENNSYLVANIA.

Billard, A., A. J. Ijspeert, and A. Martinoli (1999). A multi-robot system for adaptive exploration of a fast-changing environment: Probabilistic modeling and experimental study. *Connection Science 11*(3-4), 359–379.

Bjerknes, J. D., A. F. Winfield, and C. Melhuish (2007). An analysis of emergent taxis in a wireless connected swarm of mobile robots. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pp. 45–52. IEEE.

Bonabeau, E., M. Dorigo, and G. Theraulaz (1999). *Swarm intelligence*. Oxford.

Borzì, A. and S. Wongkaew (2015). Modeling and control through leadership of a refined flocking system. *Mathematical Models and Methods in Applied Sciences 25*(02), 255–282.

Brambilla, M., M. Dorigo, and M. Birattari (2014). Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking.

Brambilla, M., E. Ferrante, M. Birattari, and M. Dorigo (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence 7*(1), 1–41.

Brambilla, M., C. Pinciroli, M. Birattari, and M. Dorigo (2012). Property-driven design for swarm robotics. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 139–146. International Foundation for Autonomous Agents and Multiagent Systems.

Brutschy, A., L. Garattoni, M. Brambilla, G. Francesca, G. Pini, M. Dorigo, and M. Birattari (2014). The tam: abstracting complex tasks in swarm robotics research. *IRIDIA technical report series*.

Correll, N. (2007). Coordination schemes for distributed boundary coverage with a swarm of miniature robots: synthesis, analysis and experimental validation. *Online. Ph.D. thesis, Ecole Polytechnique Fédérale, Lausanne, Switzerland*.

Correll, N. (2008). Parameter estimation and optimal control of swarm-robotic systems: A case study in distributed task allocation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3302–3307. IEEE.

Correll, N. and A. Martinoli (2005). Modeling and analysis of beaconless and beacon-based policies for a swarm-intelligent inspection system. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2477–2482. IEEE.

Correll, N. and A. Martinoli (2007a). Modeling and optimization of a swarm-intelligent inspection system. In *Distributed Autonomous Robotic Systems 6*, pp. 369–378. Springer.

Correll, N. and A. Martinoli (2007b). Modeling self-organized aggregation in a swarm of miniature robots. In *IEEE 2007 International Conference on Robotics and Automation Workshop on Collective Behaviors inspired by Biological and Biochemical Systems*.

Couceiro, M. S., P. A. Vargas, R. P. Rocha, and N. M. Ferreira (2014). Benchmark of swarm robotics distributed techniques in a search task. *Robotics and Autonomous Systems 62*(2), 200 – 213.

Crespi, V., A. Galstyan, and K. Lerman (2008). Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots 24*(3), 303–313.

Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application.

de Oca, M. A. M., E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, and M. Dorigo (2011). Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making. *Swarm Intelligence 5*(3-4), 305–327.

Deloach, S. A., W. H. Oyenan, and E. T. Matson (2008). A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems 16*(1), 13–56.

Dixon, C., A. Winfield, and M. Fisher (2011). Towards temporal verification of emergent behaviours in swarm robotic systems. In *Towards Autonomous Robotic Systems*, pp. 336–347. Springer.

Ducatelle, F., A. Brutschy, A. Campo, T. Baaboura, F. Mondada, M. Brambilla, T. Stirling, V. Trianni, C. Pinciroli, S. Nolfi, et al. (2012). Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. Technical report.

El Zoghby, N., V. Loscri, E. Natalizio, V. Cherfaoui, et al. (2014). Robot cooperation and swarm intelligence. *Wireless Sensor and Robot Networks: From Topology Control to Communication Aspects*.

Evans, W. C., G. Mermoud, and A. Martinoli (2010). Comparing and modeling distributed control strategies for miniature self-assembling robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1438–1445. IEEE.

Francesca, G., M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari (2014). Automode: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*.

Francesca, G., M. Brambilla, V. Trianni, M. Dorigo, and M. Birattari (2012). Analysing an evolved robotic behaviour using a biological model of collegial decision making. In *From Animals to Animats 12*, pp. 381–390. Springer.

Galstyan, A., T. Hogg, and K. Lerman (2005, June). Modeling and mathematical analysis of swarms of microscopic robots. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pp. 201–208.

Galstyan, A. and K. Lerman (2005). Analysis of a stochastic model of adaptive task allocation in robots. In *Engineering Self-Organising Systems*, pp. 167–179. Springer.

Garnier, S., J. Gautrais, and G. Theraulaz (2007). The biological principles of swarm intelligence. *Swarm Intelligence 1*(1), 3–31.

Gazi, V. (2013). On lagrangian dynamics based modeling of swarm behavior. *Physica D: Nonlinear Phenomena 260*, 159–175.

Gazi, V. and B. Fidan (2007). Coordination and control of multi-agent dynamic systems: Models and approaches.

Gazi, V. and K. Passino (2003, April). Stability analysis of swarms. *Automatic Control, IEEE Transactions on 48*(4), 692–697.

Gazi, V. and K. M. Passino (2004). Stability analysis of social foraging swarms. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34*(1), 539–557.

Gerkey, B., R. T. Vaughan, and A. Howard (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, Volume 1, pp. 317–323.

Gerrard, C. E., J. McCall, C. Macleod, and G. M. Coghill (2013). Artificial chemistry approach to exploring search spaces using artificial reaction network agents. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 1201–1208. IEEE.

Gjondrekaj, E., M. Loreti, R. Pugliese, F. Tiezzi, C. Pinciroli, M. Brambilla, M. Birattari, and M. Dorigo (2012). Towards a formal verification methodology for collective robotic systems. In *Formal Methods and Software Engineering*, pp. 54–70. Springer.

Gomes, J. and A. L. Christensen (2013). Generic behaviour similarity measures for evolutionary swarm robotics. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pp. 199–206. ACM.

Goswami, D. and H. Hamann (2014). Investigation of a collective decision making system of different neighbourhood-size based on hyper-geometric distribution. *arXiv preprint arXiv:1410.5738*.

Grayson, S. Search & rescue using multi-robot systems.

Guo, H., Y. Meng, and Y. Jin (2010). Analysis of local communication load in shape formation of a distributed morphogenetic swarm robotic system. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8. IEEE.

Gustafson, S. and D. Gustafson (2004). Scaling issues with robot search and tagging. In *RoboSphere 2004 Workshop, NASA Ames Research Center, USA*.

Gustafson, S. and D. A. Gustafson (2006). Issues in the scaling of multi-robot systems for general problem solving. *Autonomous Robots 20*(2), 125–136.

Hamann, H. (2012). Towards swarm calculus: universal properties of swarm performance and collective decisions. In *Swarm Intelligence*, pp. 168–179. Springer.

Hamann, H. (2013). Towards swarm calculus: urn models of collective decisions and universal properties of swarm performance. *Swarm Intelligence 7*(2-3), 145–172.

Hamann, H., B. Meyer, T. Schmickl, and K. Crailsheim (2010). A model of symmetry breaking in collective decision-making. In *From Animals to Animats 11*, pp. 639–648. Springer.

Hamann, H., T. Schmickl, H. Wörn, and K. Crailsheim (2012). Analysis of emergent symmetry breaking in collective decision making. *Neural Computing and Applications 21*(2), 207–218.

Hamann, H., G. Valentini, Y. Khaluf, and M. Dorigo (2014). Derivation of a micro-macro link for collective decision-making systems. In T. Bartz-Beielstein, J. Branke, B. Filipiŏ, and J. Smith (Eds.), *Parallel Problem Solving from Nature - PPSN XIII*, Volume 8672 of *Lecture Notes in Computer Science*, pp. 181–190. Springer International Publishing.

Hamann, H. and H. Wörn (2007a). An analytical and spatial model of foraging in a swarm of robots. In *Swarm Robotics*, pp. 43–55. Springer.

Hamann, H. and H. Wörn (2007b). A space- and time-continuous model of self-organizing robot swarms for design support. In *SASO '07: Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems*, Washington, DC, USA, pp. 23. IEEE Computer Society.

Hamann, H. and H. Wörn (2008). A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intelligence 2*(2-4), 209–239.

Hamann, H., H. Wörn, K. Crailsheim, and T. Schmickl (2008). Spatial macroscopic models of a bio-inspired robotic swarm algorithm. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1415–1420. IEEE.

Hashmi, S. and W. Mahmood (2006). A matrix based approach for modeling robotic swarm behavior. In *IC-AI*, pp. 433–438. Citeseer.

Hereford, J. M. (2010). Analysis of a new swarm search algorithm based on trophallaxis. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8. IEEE.

Higgins, F., A. Tomlinson, and K. M. Martin (2009). Survey on security challenges for swarm robotics. In *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*, pp. 307–312. IEEE.

Hosking, M. and F. Sahin (2010). Testability of a swarm robot using a system of systems approach and discrete event simulation. In *System of Systems Engineering (SoSE), 2010 5th International Conference on*, pp. 1–6. IEEE.

Hsieh, M. A. and T. W. Mather (2012). Distributed aggregation in the presence of uncertainty: A statistical physics approach. In *2012 AAAI Spring Symposium Series*.

Huh, S., S. Hong, and J. Lee (2013). Energy-efficient distributed programming model for swarm robot. In *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*, pp. 300–305. IEEE.

Hunter, J., F. Raimondi, N. Rungta, and R. Stocker (2013). A synergistic and extensible framework for multi-agent system verification. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 869–876. International Foundation for Autonomous Agents and Multiagent Systems.

Ijspeert, A. J., A. Martinoli, A. Billard, and L. M. Gambardella (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots 11*(2), 149–171.

Ingenieurwissenschaften, D. D., D. inf Heiko Hamann, E. Gutachter, P. D. ing, H. Wörn, Z. Gutachter, P. D. ing, and U. D. Hanebeck (2008). Space-time continuous models of swarm robotic systems: Supporting global-to-local programming zur erlangung des akademischen grades eines.

Jain, S., M. Sawlani, and V. K. Chandwani (2010). Ad-hoc swarm robotics optimization in grid based navigation. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pp. 1553–1558. IEEE.

Jantz, S. D., K. L. Doty, J. A. Bagnell, and I. R. Zapata (1997). Kinetics of robotics: The development of universal metrics in robotic swarms. In *in Florida Conference on Recent Advances in Robotics*.

Jin, D. and L. Gao (2006). Stability analysis of swarm based on double integrator model. In *Computational Intelligence and Bioinformatics*, pp. 201–210. Springer.

Jin, D. and L. Gao (2008). Stability analysis of a double integrator swarm model related to

position and velocity. *Transactions of the Institute of Measurement and Control 30*(3-4), 275–293.

Jones, C. V., M. Matarić, G. Sukhatme, K. Lerman, S. Koenig, and U. Mitra (2004). A formal design methodology for coordinated multi-robot systems.

Kazadi, S. (2009). Model independence in swarm robotics. *International Journal of Intelligent Computing and Cybernetics 2*(4), 672–694.

Kerr, W., D. Spears, W. Spears, and D. Thayer (2005). Two formal gas models for multi-agent sweeping and obstacle avoidance. In *Formal Approaches to Agent-Based Systems*, pp. 111–130. Springer.

Kettler, A. and H. Wörn (2011a). A framework for boltzmann-type models of robotic swarms. In *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, pp. 1–8. ID: 1.

Kettler, A. and H. Wörn (2011b). Sorting boxes with a robotic swarm: An analysis by means of experiment, simulation and model. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pp. 865–870. IEEE.

Khaluf, Y., M. Birattari, and H. Hamann (2014). A swarm robotics approach to task allocation under soft deadlines and negligible switching costs. In A. del Pobil, E. Chinellato, E. Martinez-Martin, J. Hallam, E. Cervera, and A. Morales (Eds.), *From Animals to Animats 13*, Volume 8575 of *Lecture Notes in Computer Science*, pp. 270–279. Springer International Publishing.

Khaluf, Y., M. Birattari, and F. Rammig (2013). Probabilistic analysis of long-term swarm performance under spatial interferences. In *Theory and Practice of Natural Computing*, pp. 121–132. Springer.

Khaluf, Y., M. Pace, F. Rammig, and M. Dorigo (2013a). Integrals of markov processes with application to swarm robotics modelling. *IEEE Transactions on Robotics, page in revision*.

Khaluf, Y., M. Pace, F. Rammig, and M. Dorigo (2013b). Integrals of markov processes with application to swarm robotics modelling. *IEEE Transactions on Robotics, page in revision*.

Klavins, E. (2004). A language for modeling and programming cooperative control systems. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Volume 4, pp. 3403–3410. IEEE.

Kloetzer, M. and C. Belta (2006). Hierarchical abstractions for robotic swarms. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 952–957. IEEE.

Koenig, N. and A. Howard (2004, Sept). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, Volume 3, pp. 2149–2154 vol.3.

Konur, S., C. Dixon, and M. Fisher (2010). Formal verification of probabilistic swarm behaviours. In *Swarm Intelligence*, pp. 440–447. Springer.

Korb, J. (2003). Thermoregulation and ventilation of termite mounds. *Naturwissenschaften 90*(5), 212–219.

Kouvaros, P. and A. Lomuscio (2013). Automatic verification of parameterised multi-agent systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, Richland, SC, pp. 861–868. International Foundation for Autonomous Agents and Multiagent Systems.

Kouvaros, P. and A. Lomuscio (2015). A counter abstraction technique for the verification of robot swarms.

Kumar, G. P., A. Buffin, T. P. Pavlic, S. C. Pratt, and S. M. Berman (2013). A stochastic hybrid system model of collective transport in the desert ant aphaenogaster cockerelli. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pp. 119–124. ACM.

Kumar, M., D. Milutinovic, and D. P. Garg (2008). Role of stochasticity in self-organization of robotic swarms. In *American Control Conference, 2008*, pp. 123–128. IEEE.

Lancaster, J. P. and D. A. Gustafson (2010). Predicting performance in robotic search and tag. In *Conference on Artificial Neural Networks in Engineering (ANNIE 2010)*.

Lancaster, J. P. and D. A. Gustafson (2013). Predicting the behavior of robotic swarms in search and tag tasks. *Procedia Computer Science 20*(0), 77 – 82. Complex Adaptive Systems.

Lerman, K. (2004). A model of adaptation in collaborative multi-agent systems. *Adaptive Behavior 12*(3-4), 187–197.

Lerman, K. and A. Galstyan (2001). A general methodology for mathematical analysis of multi-agent systems. *ISI-TR-529, USC Information Sciences Institute, Marina del Rey, CA*.

Lerman, K. and A. Galstyan (2002). Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots 13*(2), 127–141.

Lerman, K. and A. Galstyan (2003). Macroscopic analysis of adaptive task allocation in robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Volume 2, pp. 1951–1956. IEEE.

Lerman, K. and A. Galstyan (2004). Two paradigms for the design of artificial collectives. *Kagan Tumer and David Wolpert (Eds.) Collectives and Design of Complex Systems*, 231–256.

Lerman, K., A. Galstyan, A. Martinoli, and A. Ijspeert (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life 7*(4), 375–393.

Lerman, K., C. Jones, A. Galstyan, and M. J. Matarić (2006). Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research 25*(3), 225–241.

Lerman, K., A. Martinoli, and A. Galstyan (2004). A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm Robotics Workshop at the Eight Int. Conference on the Simulation of Adaptive Behavior SAB-04*. EPFL. E. Sahin and W. Spears, editors,, Los Angeles, CA. Lecture Notes in Computer Science (2004).

Lerman, K., A. Martinoli, and A. Galstyan (2005). A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm robotics*, pp. 143–152. Springer.

Lerman, K., M. Matarić, and A. Galstyan (2005). Mathematical modeling of large multi-agent systems. Technical report, DTIC Document.

Lindsay, J., S. Givigi, G. Pieris, G. Fusina, and G. Labonte (2012). Experimental validation of swarms of robots. In *Systems Conference (SysCon), 2012 IEEE International*, pp. 1–5. IEEE.

Liu, W. (2008). *Design and Modelling of Adaptive Foraging in Swarm Robotic Systems*. Ph. D. thesis, Faculty of Environment and Technology, University of the West of England, Bristol.

Liu, W. and A. F. Winfield (2010). Modeling and optimization of adaptive foraging in swarm robotic systems. *The International Journal of Robotics Research 29*(14), 1743–1760.

Liu, W., A. F. Winfield, and J. Sa (2007). Modelling swarm robotic systems: A case study in collective foraging. *Towards Autonomous Robotic Systems (TAROS 07)*, 25–32.

Mahmood, W. and S. Hashmi (2006). Modeling complex emergent discrete event systems: a case study in robotic swarm motion. In *Proceedings of the 8th WSEAS international conference on Automatic control, modeling & simulation*, pp. 117–122. World Scientific and Engineering Academy and Society (WSEAS).

Martinoli, A. (1999). Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies. *Unpublished doctoral manuscript, EPFL Ph. D. Thesis* (2069).

Martinoli, A. and K. Easton (2003a). Modeling swarm robotic systems. In *Experimental Robotics VIII*, pp. 297–306. Springer.

Martinoli, A. and K. Easton (2003b). Optimization of swarm robotic systems via macroscopic models. In *Proc. of the Second Int. Workshop on Multi-Robots Systems*, pp. 181–192.

Martinoli, A., K. Easton, and W. Agassounon (2004). Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *The International Journal of Robotics Research 23*(4-5), 415–436.

Martinoli, A., A. J. Ijspeert, and L. M. Gambardella (1999). A probabilistic model for understanding and comparing collective aggregation mechanisms. In *Advances in artificial life*, pp. 575–584. Springer.

Martinoli, A., A. J. Ijspeert, and F. Mondada (1999). Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems 29*(1), 51–63.

Massink, M., M. Brambilla, D. Latella, M. Dorigo, and M. Birattari (2012). Analysing robot swarm decision-making with bio-pepa. In *Swarm Intelligence*, pp. 25–36. Springer.

Massink, M., M. Brambilla, D. Latella, M. Dorigo, and M. Birattari (2013). On the use of bio-pepa for modelling and analysing collective behaviours in swarm robotics. *Swarm Intelligence 7*(2-3), 201–228.

Mather, T. W. and M. A. Hsieh (2011). Distributed robot ensemble control for deployment to multiple sites. *Robotics: Science and Systems VII*.

Mendez, L., S. N. Givigi, H. M. Schwartz, A. Beaulieu, G. Pieris, and G. Fusina (2012). Validation of swarms of robots: theory and experimental results. In *System of Systems Engineering (SoSE), 2012 7th International Conference on*, pp. 332–337. IEEE.

Mermoud, G., U. Upadhyay, W. C. Evans, and A. Martinoli (2014). Top-down vs. bottom-up model-based methodologies for distributed control: A comparative experimental study. In *Experimental Robotics*, pp. 615–629. Springer.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems 1*(1), 39–42.

Mlot, N. J., C. A. Tovey, and D. L. Hu (2011). Fire ants self-assemble into waterproof rafts to survive floods. *Proceedings of the National Academy of Sciences 108*(19), 7669–7673.

Mohan, Y. and S. Ponnambalam (2009). An extensive review of research in swarm robotics. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 140–145. IEEE.

Mueller, U. G. and N. Gerardo (2002). Fungus-farming insects: multiple origins and diverse evolutionary histories. *Proceedings of the National Academy of Sciences 99*(24), 15247–15249.

Muniganti, P. and A. O. Pujol (2010). A survey on mathematical models of swarm robotics. In *Workshop of Physical Agents*.

Nakagaki, T. (2001). Smart behavior of true slime mold in a labyrinth. *Research in Microbiology 152*(9), 767–770.

Nakagaki, T., H. Yamada, and Á. Tóth (2000). Intelligence: Maze-solving by an amoeboid organism. *Nature 407*(6803), 470–470.

Neumann, J. v. and A. W. Burks (1966). Theory of self-reproducing automata.

Ohkura, K., T. Yasuda, and Y. Matsumura (2011). Analyzing macroscopic behavior in a swarm robotic system based on clustering. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pp. 356–361. IEEE.

Pace, M., M. Birattari, and M. Dorigo (2013a). Random finite set modeling of swarm robotics systems and roboplasms: Fundamentals. *IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep*.

Pace, M., M. Birattari, and M. Dorigo (2013b). The swarm/potential model: Modeling robotics swarms with measure-valued recursions associated to random finite sets. *IEEE Transactions on Robotics, page submitted*.

Parker, C. A., H. Zhang, and C. R. Kube (2003). Blind bulldozing: multiple robot nest construction. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Volume 2, pp. 2010–2015. IEEE.

Parker, L. E. (1998). Alliance: An architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on 14*(2), 220–240.

Pimenta, L. C., G. A. Pereira, N. Michael, R. C. Mesquita, M. M. Bosque, L. Chaimowicz, and V. Kumar (2013). Swarm coordination based on smoothed particle hydrodynamics technique.

Pinciroli, C., V. Trianni, R. OGrady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, et al. (2012). Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence 6*(4), 271–295.

Prokopenko, M. (2013). Information dynamics at the edge of chaos: Measures, examples, and principles. In *2013 IEEE Symposium on Artificial Life (ALIFE)*, pp. 148–152.

Prorok, A., N. Correll, and A. Martinoli (2011). Multi-level spatial modeling for stochastic distributed robotic systems. *The International Journal of Robotics Research 30*(5), 574–589.

Ranjbar-Sahraei, B., G. Weiss, and K. Tuyls (2013). A macroscopic model for multi-robot stigmergic coverage. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1233–1234. International Foundation for Autonomous Agents and Multiagent Systems.

Rebguns, A., R. Anderson-Sprecher, D. Spears, W. Spears, and A. Kletsov (2008). Using scouts to predict swarm success rate. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pp. 1–8. IEEE.

Reina, A., M. Dorigo, and V. Trianni (2014). Towards a cognitive design pattern for collective decision-making. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Sttzle (Eds.), *Swarm Intelligence*, Volume 8667 of *Lecture Notes in Computer Science*, pp. 194–205. Springer International Publishing.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, Volume 21, pp. 25–34. ACM.

Rouff, C., W. Truszkowski, J. Rash, and M. Hinchey (2003). Formal approaches to intelligent swarms. In *Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard*, pp. 51–57. IEEE.

Roy, S., S. Biswas, and S. S. Chaudhuri (2014). Nature-inspired swarm intelligence and its applications. *International Journal of Modern Education and Computer Science (IJMECS) 6*(12), 55.

Rubenstein, M., A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal (2013). Collective transport of complex objects by simple robots: theory and experiments. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 47–54. International Foundation for Autonomous Agents and Multiagent Systems.

Sargeant, I. and A. Tomlinson (2013). Modelling malicious entities in a robotic swarm. In *Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd*, pp. 7B1–1. IEEE.

Schmickl, T., H. Hamann, H. Wörn, and K. Crailsheim (2009). Two different approaches to a macroscopic model of a bio-inspired robotic swarm. *Robotics and Autonomous Systems 57*(9), 913–921.

Shi, Z., J. Tu, Q. Zhang, L. Liu, and J. Wei (2012). A survey of swarm robotics system. In *Advances in Swarm Intelligence*, pp. 564–572. Springer.

Soysal, O. and E. Şahin (2007). A macroscopic model for self-organized aggregation in swarm robotic systems. In *Swarm robotics*, pp. 27–42. Springer.

Stanislaw, H. and N. Todorov (1999). Calculation of signal detection theory measures. *Behavior research methods, instruments, & computers 31*(1), 137–149.

Tarapore, D., A. L. Christensen, P. U. Lima, and J. Carneiro (2013). Abnormality detection in multiagent systems inspired by the adaptive immune system. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, Richland, SC, pp. 23–30. International Foundation for Autonomous Agents and Multiagent Systems.

Tarapore, D., P. U. Lima, J. Carneiro, and A. L. Christensen (2015). To err is robotic, to tolerate immunological: fault detection in multirobot systems. *Bioinspiration & Biomimetics 10*(1), 016014.

Taylor-King, J. P., B. Franz, C. A. Yates, and R. Erban (2014). Mathematical modelling of turning delays in swarm robotics. *arXiv preprint arXiv:1401.7612*.

Vain, J., T. Tammet, A. Kuusik, and S. Juurik (2008). Towards scalable proofs of robot swarm dependability. In *Electronics Conference, 2008. BEC 2008. 11th International Biennial Baltic*, pp. 199–202. IEEE.

Valentini, G., M. Birattari, and M. Dorigo (2013). Majority rule with differential latency: an absorbing markov chain to model consensus. In *Proceedings of the European Conference on Complex Systems 2012*, pp. 651–658. Springer.

Valentini, G., H. Hamann, and M. Dorigo (2014). Self-organized collective decision making: The weighted voter model. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, Richland, SC, pp. 45–52. International Foundation for Autonomous Agents and Multiagent Systems.

Vigelius, M., B. Meyer, and G. Pascoe (2014, 11). Multiscale modelling and analysis of collective decision making in swarm robotics. *PLoS ONE 9*(11), e111542.

Williams, K. I. (2006). *Multi-robot systems: modeling swarm dynamics and designing inspection planning algorithms*. Ph. D. thesis, California Institute of Technology.

Winfield, A. F., C. J. Harper, and J. Nembrini (2005). Towards dependable swarms and a new discipline of swarm engineering. In *Swarm robotics*, pp. 126–142. Springer.

Winfield, A. F., W. Liu, J. Nembrini, and A. Martinoli (2008). Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence 2*(2-4), 241–266.

Winfield, A. F. and J. Nembrini (2006). Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control 1*(1), 30–37.

Winfield, A. F., J. Sa, M.-C. Fernández-Gago, C. Dixon, and M. Fisher (2005). On formal specification of emergent behaviours in swarm robotic systems. *International journal of advanced robotic systems 2*(4).

Wirsing, M., M. Hölzl, M. Tribastone, and F. Zambonelli (2013). Ascens: Engineering autonomic service-component ensembles. In *Formal Methods for Components and Objects*, pp. 1–24. Springer.

Yasuda, T., K. Ohkura, N. Wada, and Y. Matsumura (2013). Behavior sequence analysis of incrementally evolving robotic swarms in a foraging task. In *System Integration (SII), 2013 IEEE/SICE International Symposium on*, pp. 790–795. IEEE.

Yasuda, T., N. Wada, K. Ohkura, and Y. Matsumura (2013). Analyzing collective behavior in evolutionary swarm robotic systems based on an ethological approach. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*, pp. 148–155. IEEE.

Zhai, S. and B. Fidan (2008). Single view depth estimation based formation control of

robotic swarms: Fundamental design and analysis. In *Control and Automation, 2008 16th Mediterranean Conference on*, pp. 1156–1161. IEEE.

Zhang, J. and S. T. Mueller (2005). A note on roc analysis and non-parametric estimate of sensitivity. *Psychometrika 70*(1), 203–212.

Zhang, Y., F. Bastani, I.-L. Yen, J. Fu, and I.-R. Chen (2008). Availability analysis of robotic swarm systems. In *Dependable Computing, 2008. PRDC'08. 14th IEEE Pacific Rim International Symposium on*, pp. 331–338. IEEE.

Zhiqi, L., X. Songdong, Z. Jianchao, Z. Jing, and Z. Guoyou (2010). An evaluation of pso-type swarm robotic search: Modeling method and controlling properties. In *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pp. 360–365. IEEE.

Zhou, J., D. Mu, F. Yang, and G. Dai (2014a, July). Labor division for swarm robotic systems with arbitrary finite number of task types. In *Information and Automation (ICIA), 2014 IEEE International Conference on*, pp. 1113–1118.

Zhou, J., D. Mu, F. Yang, and G. Dai (2014b, July). A novel approach for analysing collective dynamics of large-scale multi-robot system in task allocation. In *Information and Automation (ICIA), 2014 IEEE International Conference on*, pp. 1137–1142.

Zhu, A. and S. Yang (2010). A survey on intelligent interaction and cooperative control of multi-robot systems. *IEEE ICCA 2010*, 1812–1817.

Appendix A

Simulation Worlds

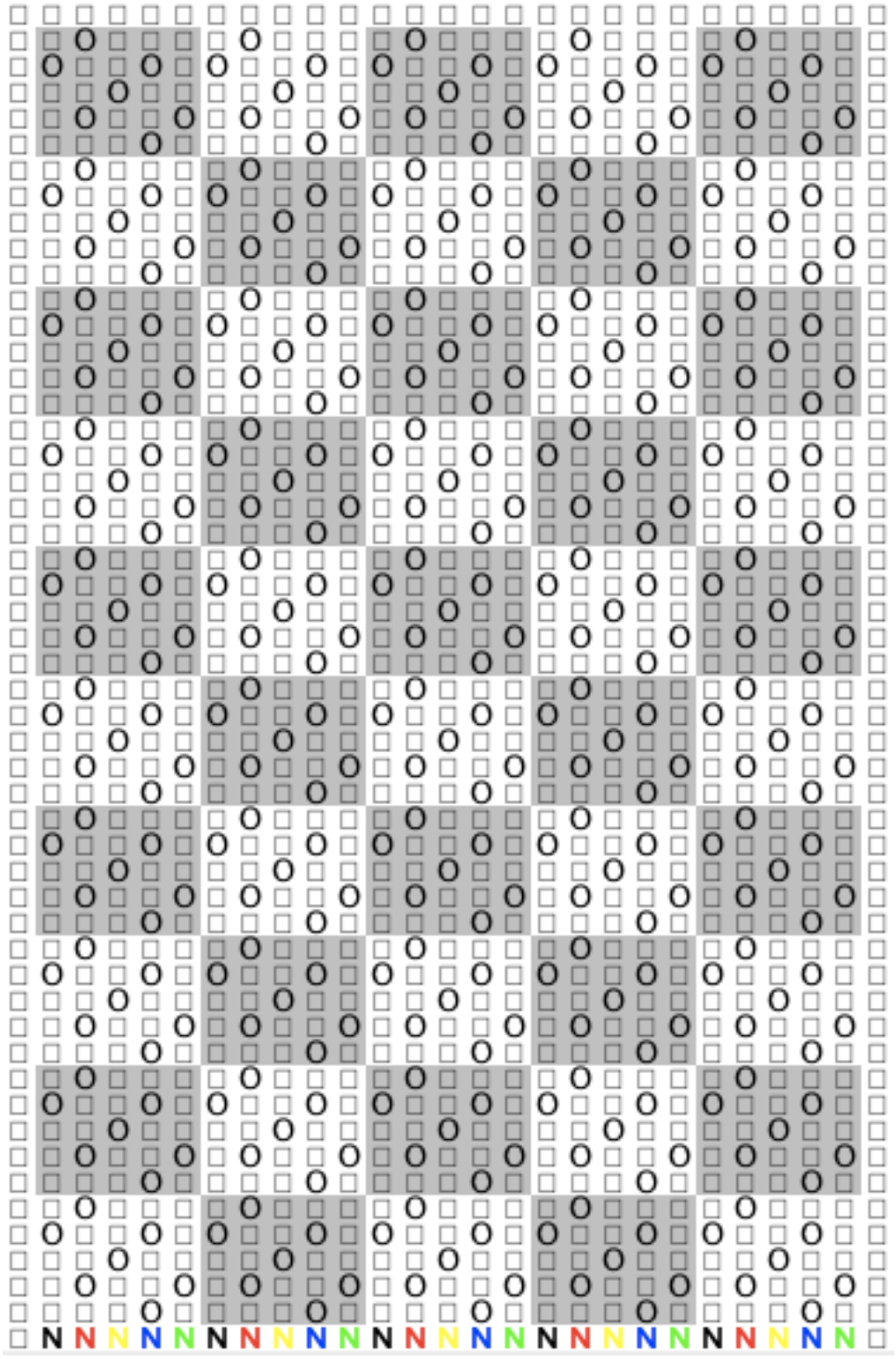

Figure A.1: *World execution "East" at step 0.*

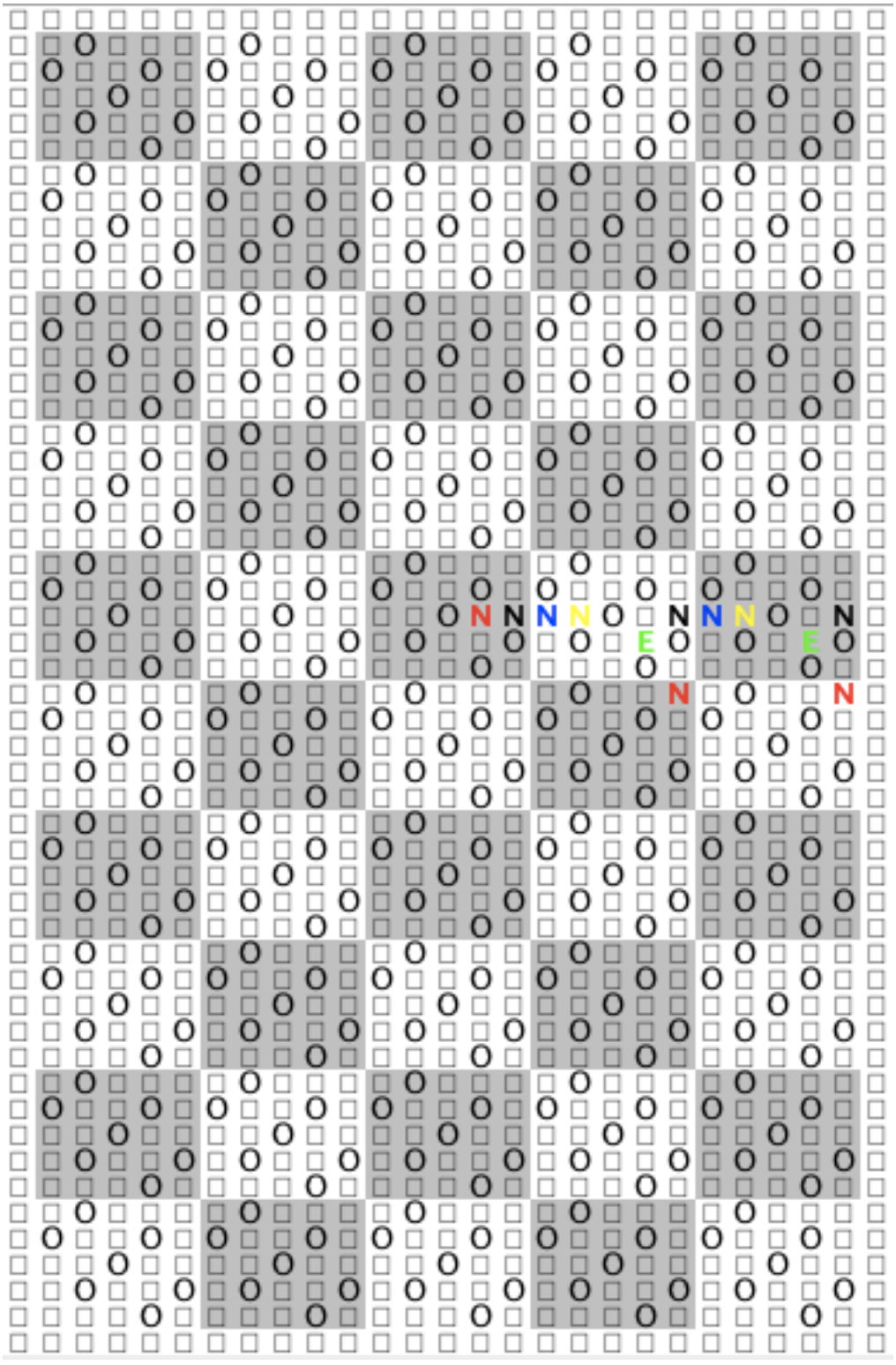

Figure A.2: *World execution "East" at step 70.*

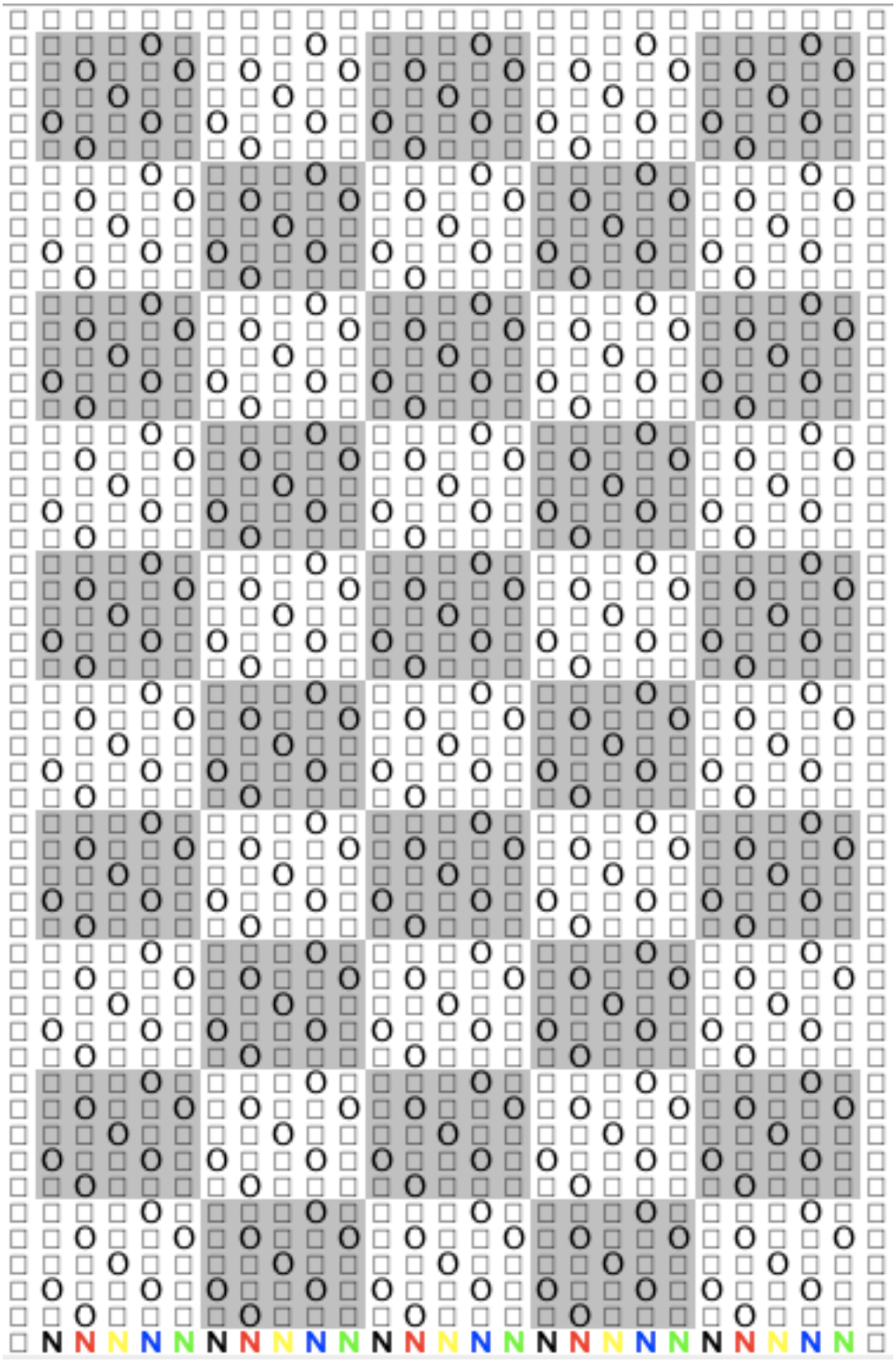

Figure A.3: *World execution "West1" at step 0.*

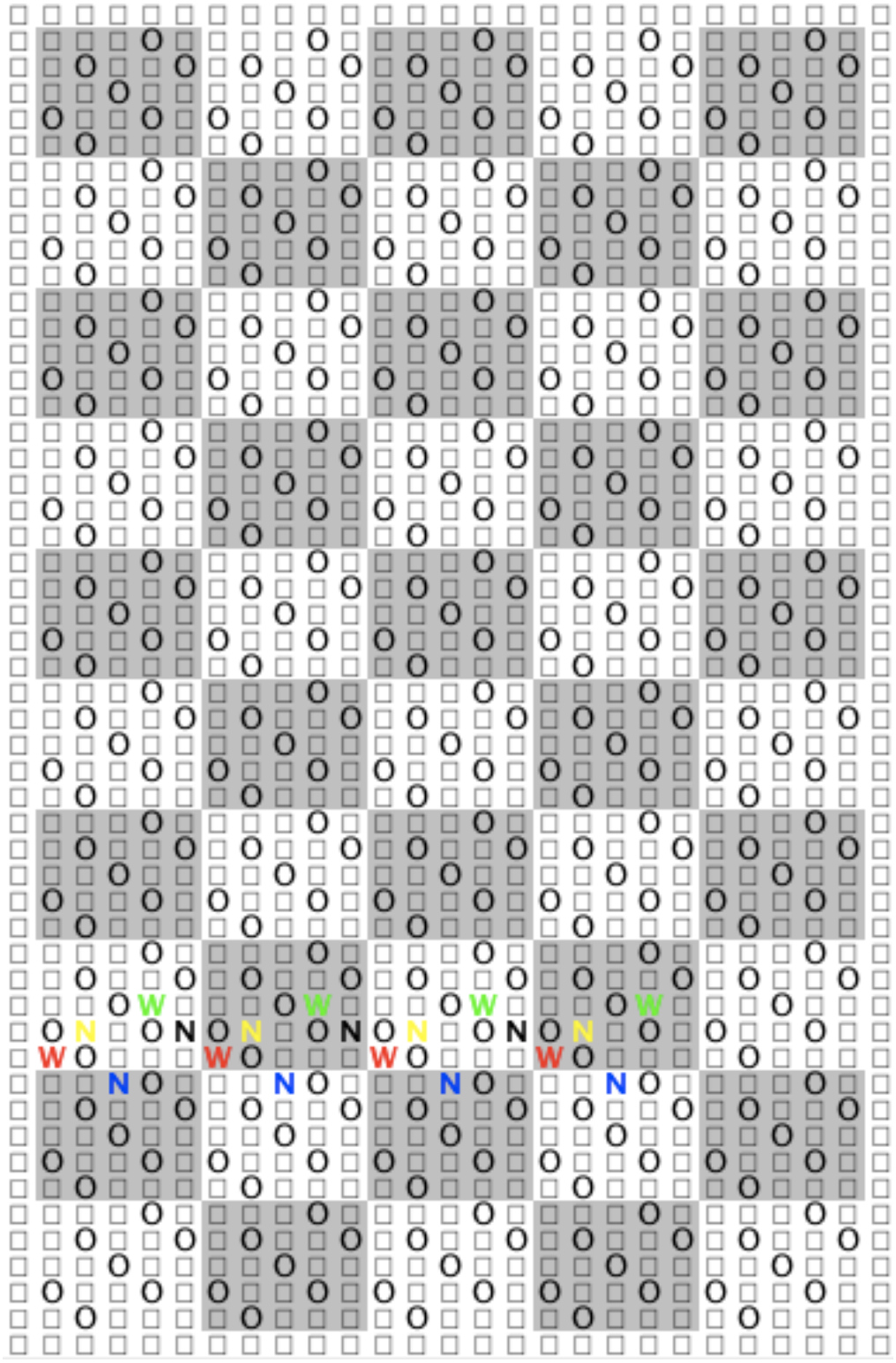

Figure A.4: *World execution "West1" at step 30.*

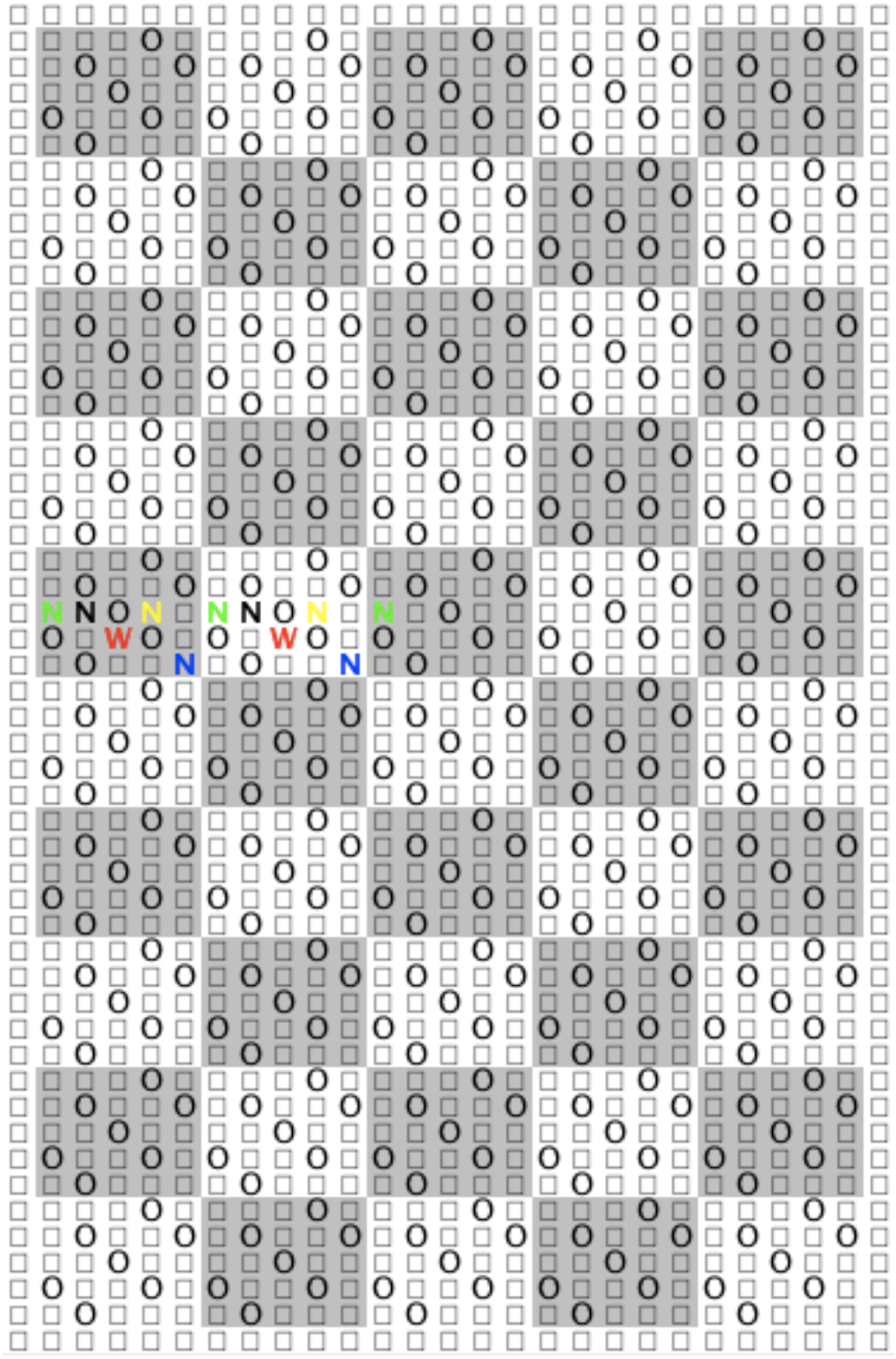

Figure A.5: *World execution "West1" at step 70.*

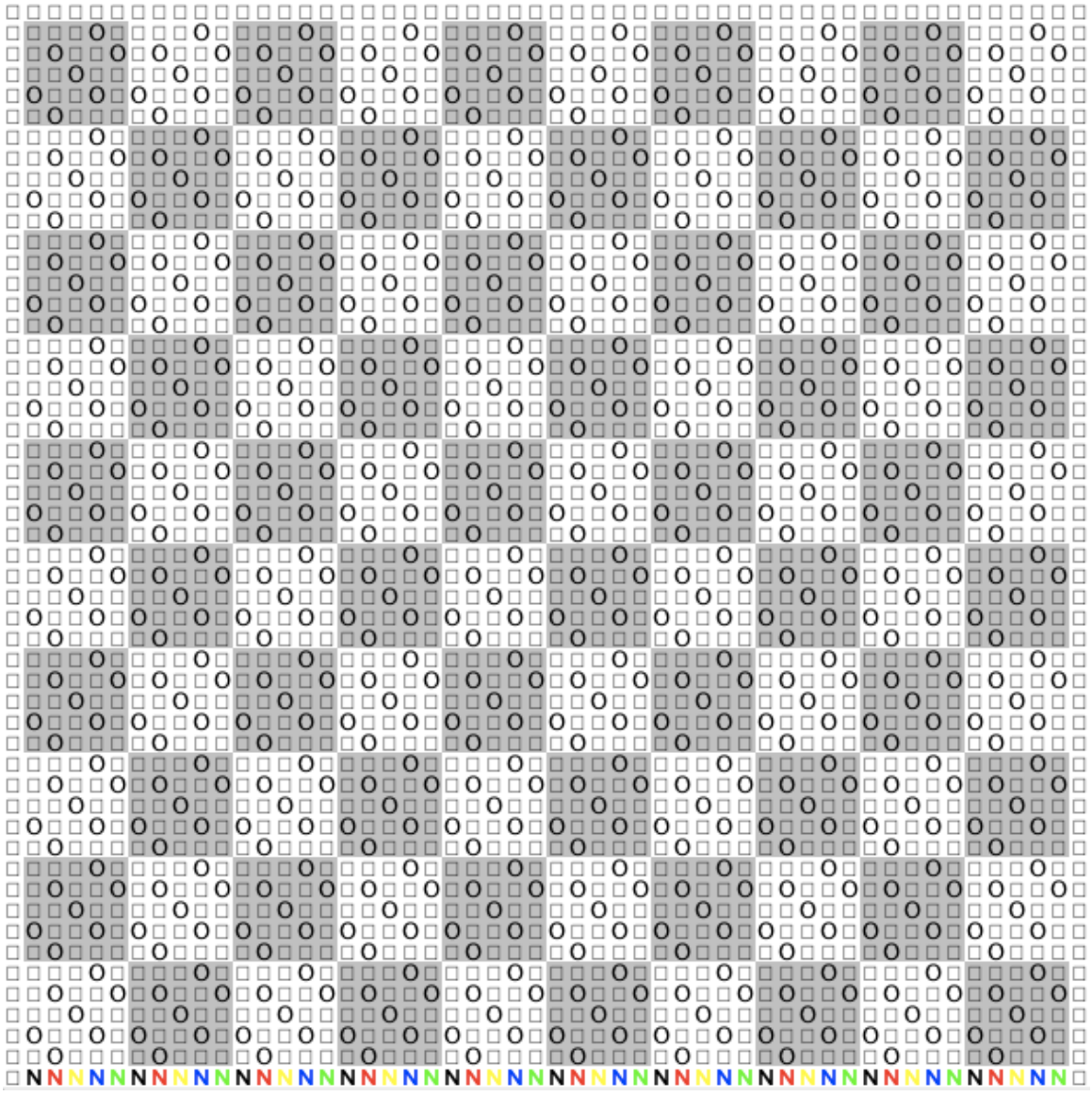

Figure A.6: *World execution "West1 Large" at step 0.*

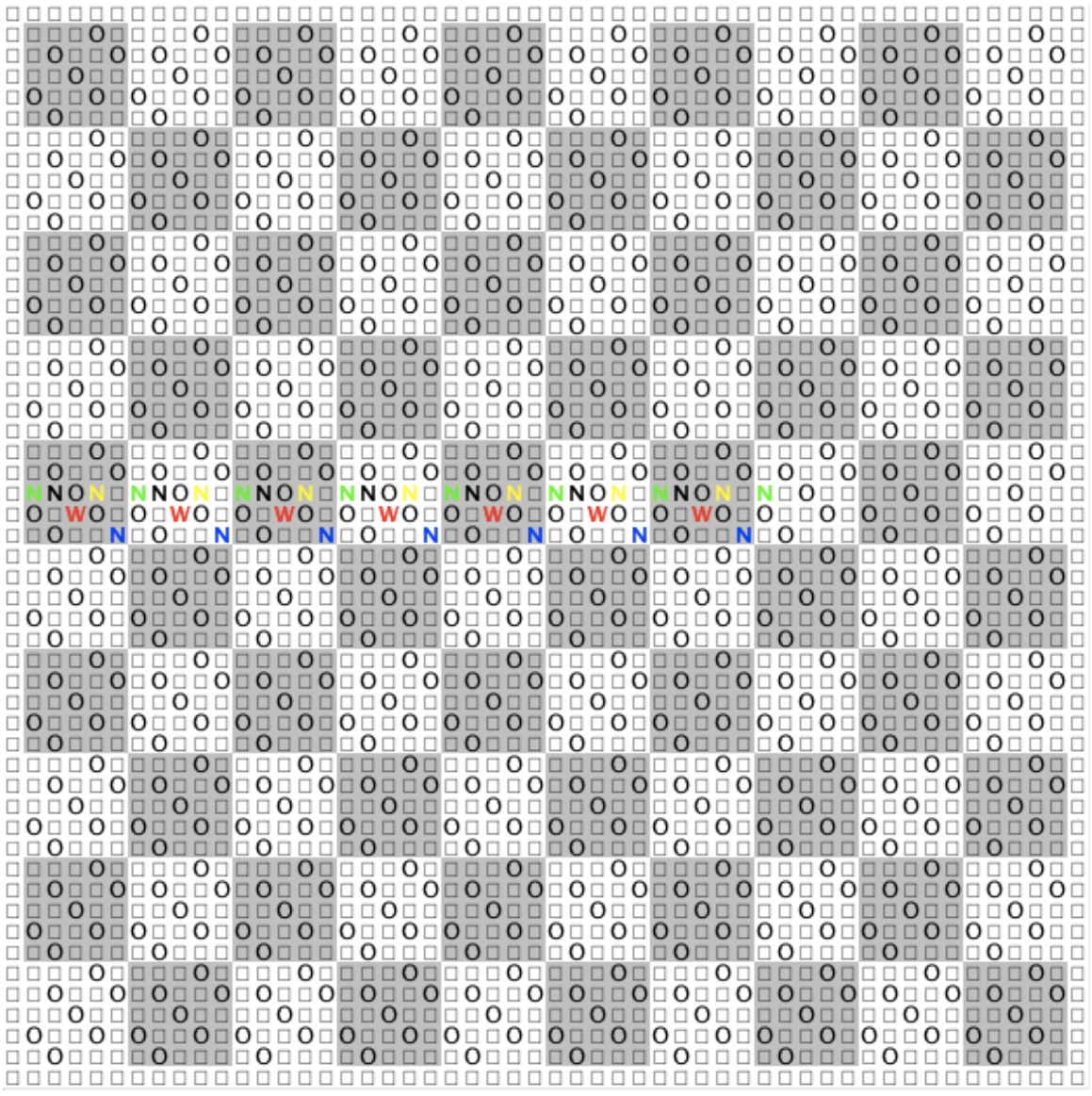

Figure A.7: *World execution "West1 Large" at step 70.*

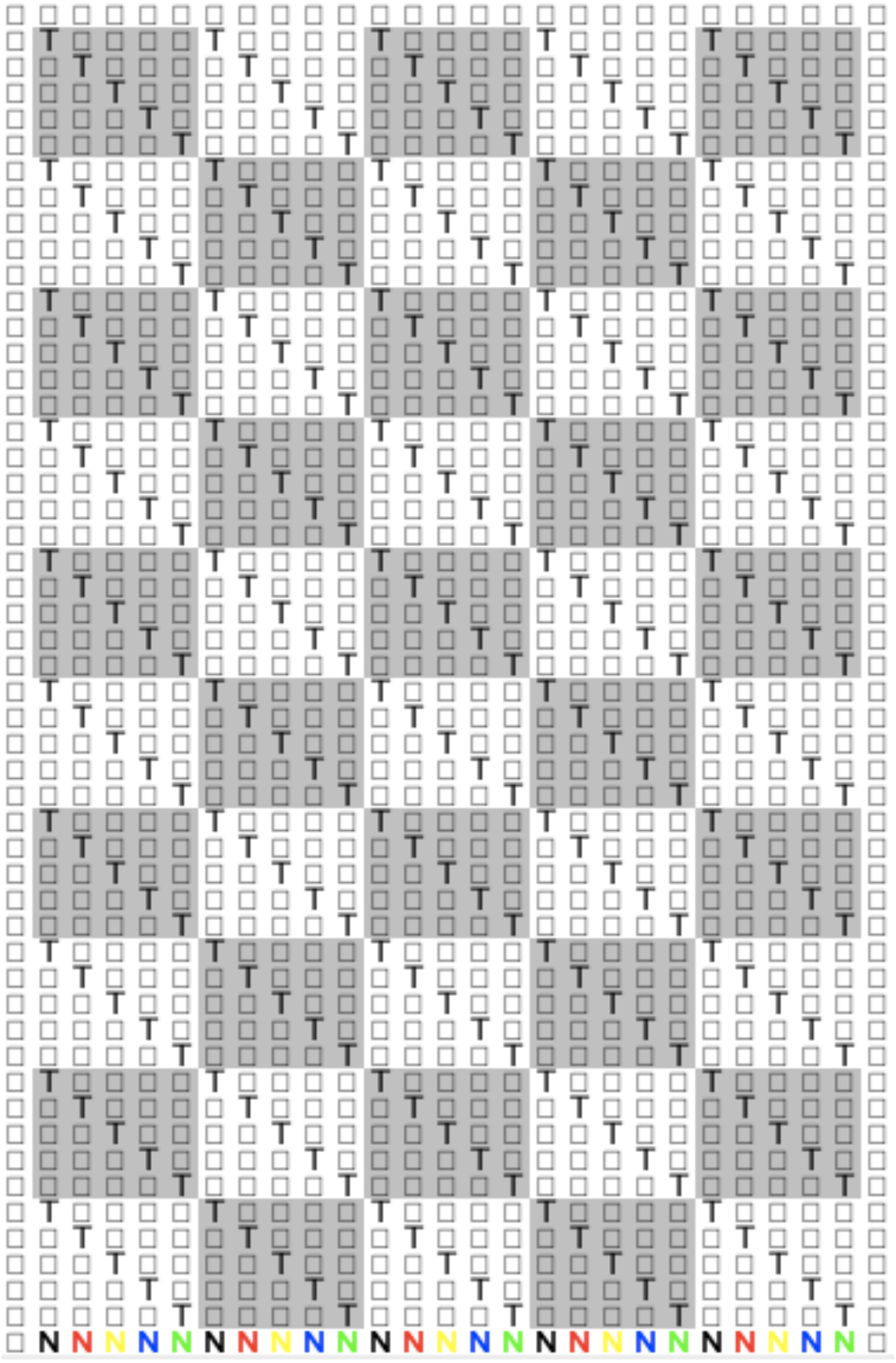

Figure A.8: *World execution "West2" at step 0.*

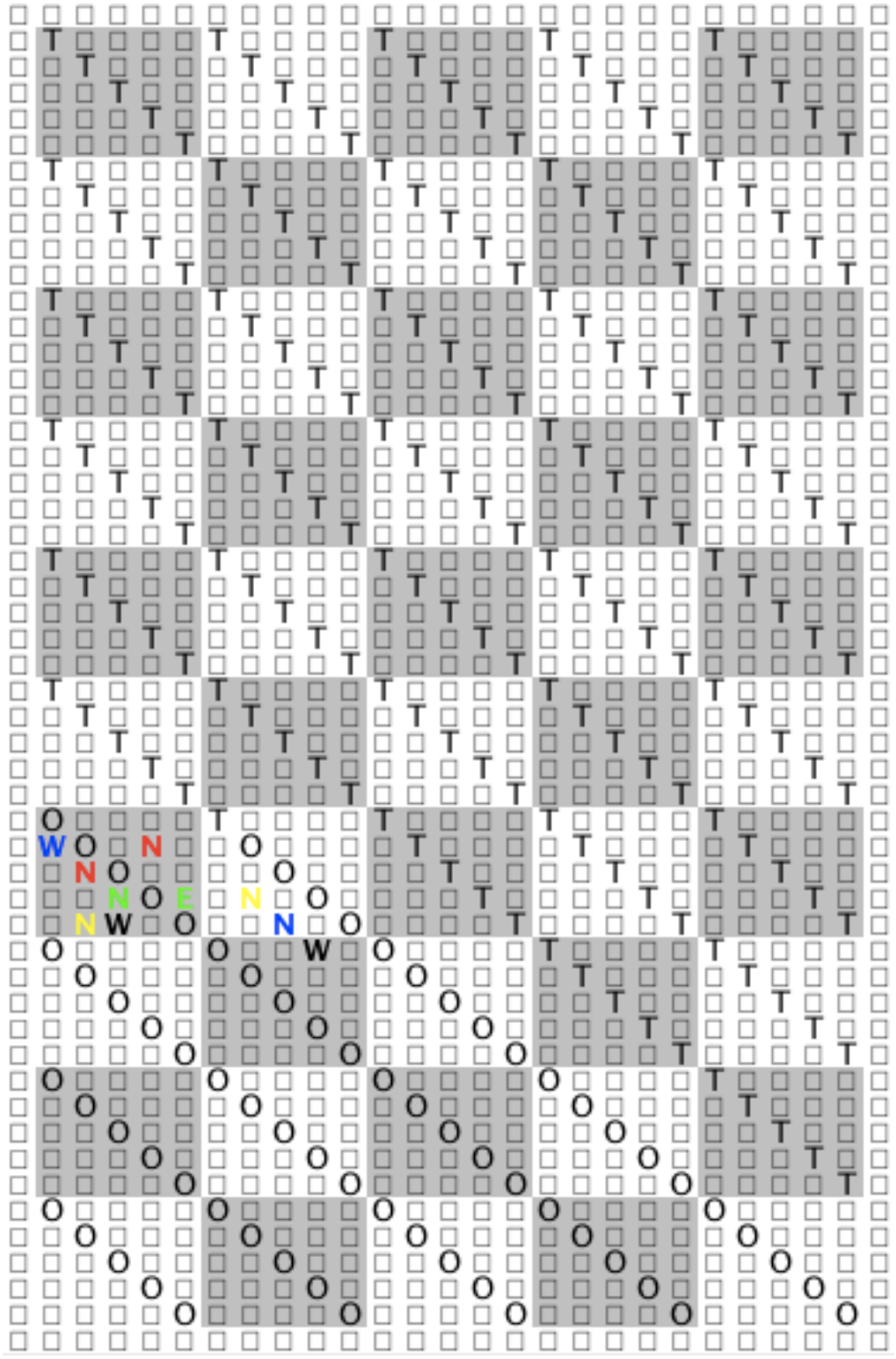

Figure A.9: *World execution "West2" at step 70.*

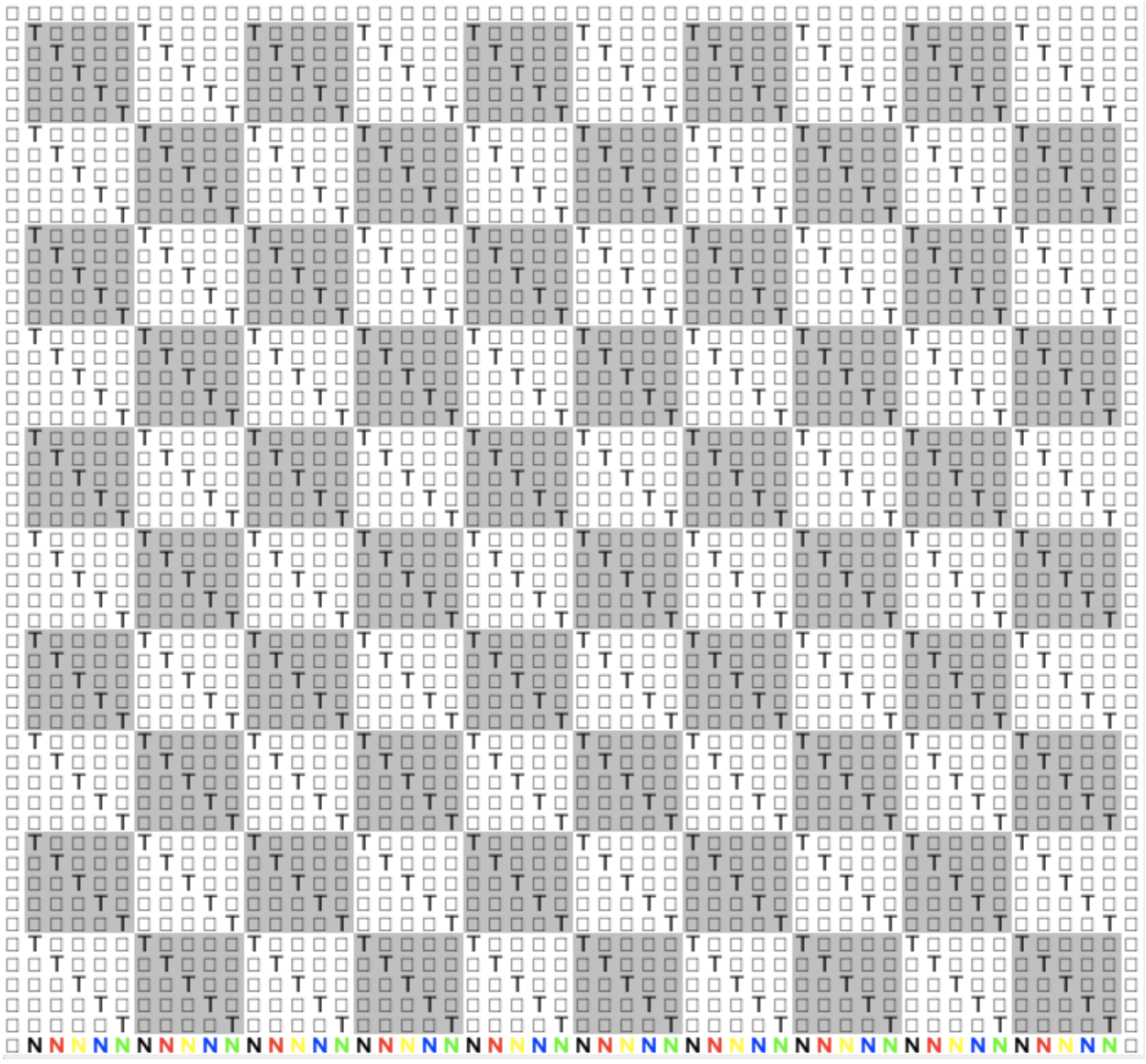

Figure A.10: *World execution "West2 Large" at step 0.*

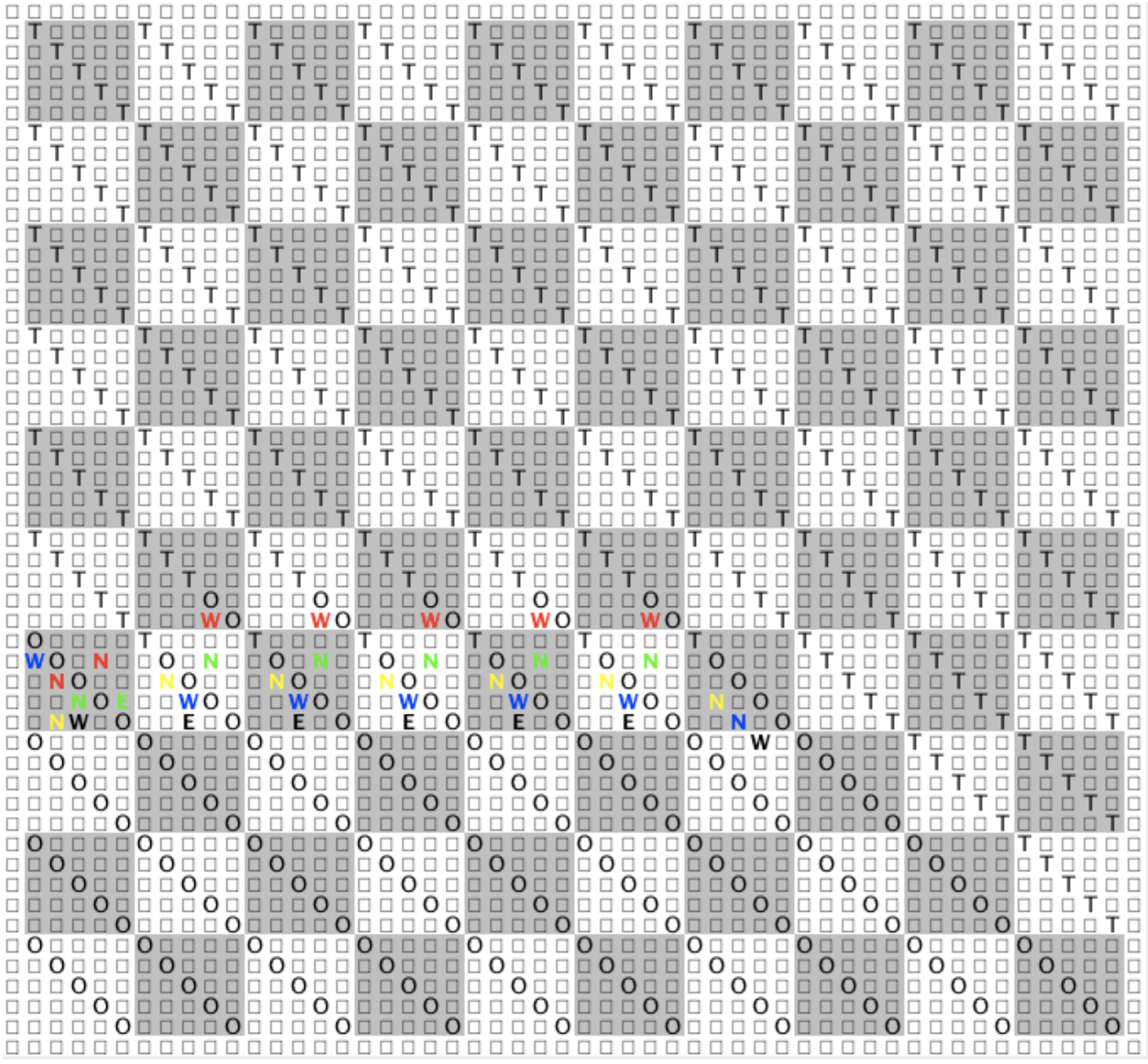

Figure A.11: *World execution "West2 Large" at step 70.*

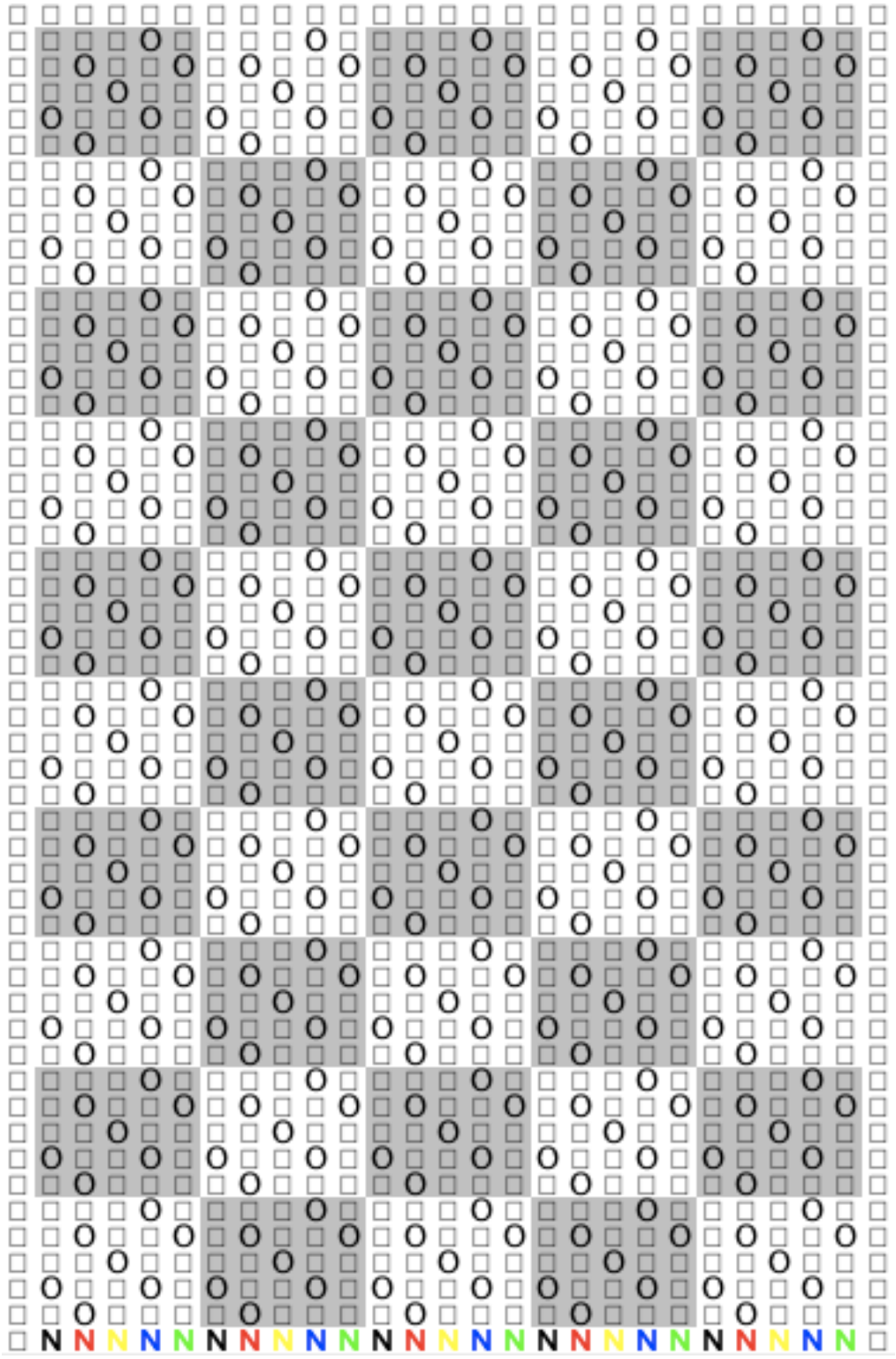

Figure A.12: *World execution "West3" at step 0.*

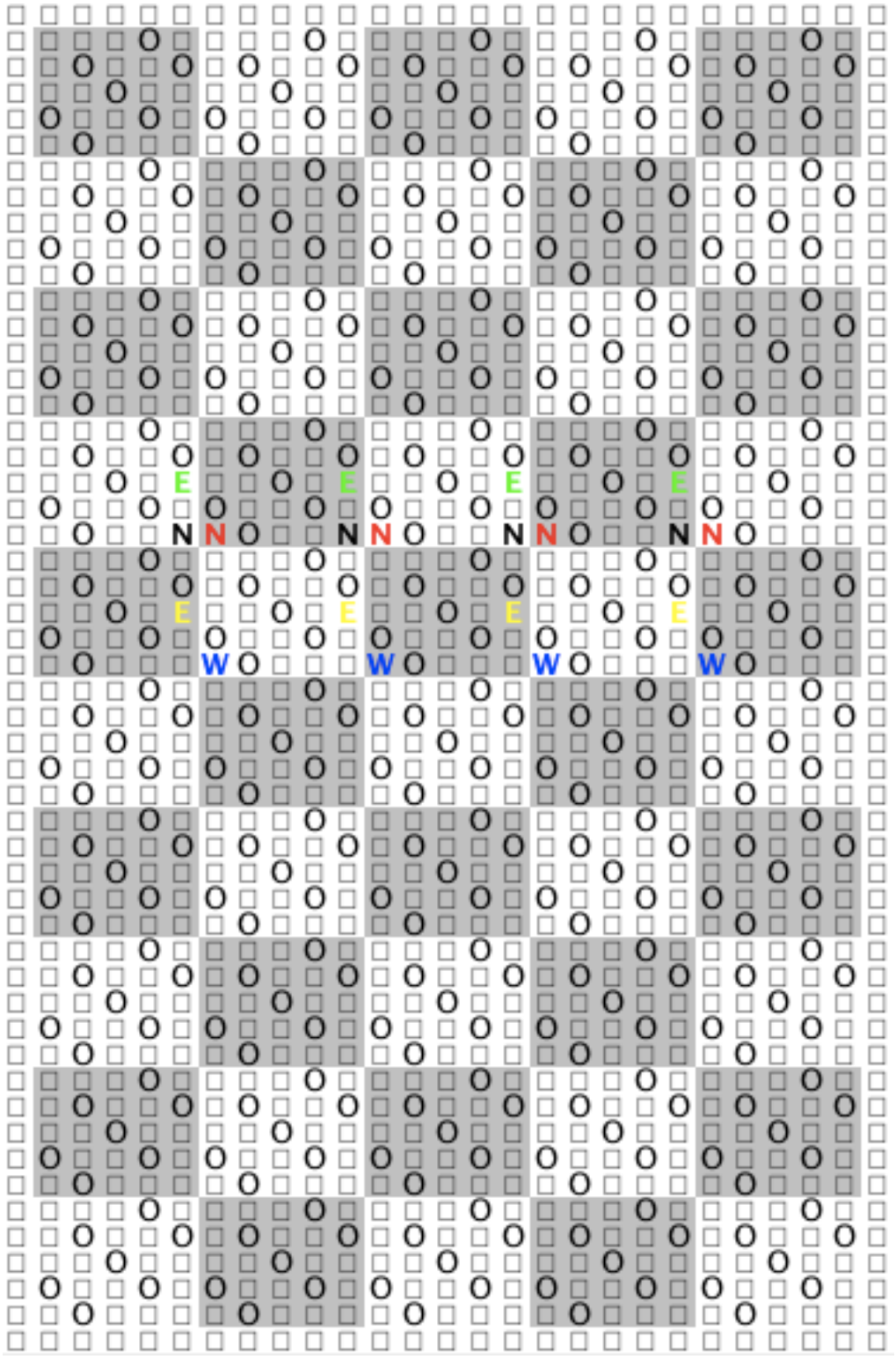

Figure A.13: *World execution "West3" at step 70.*

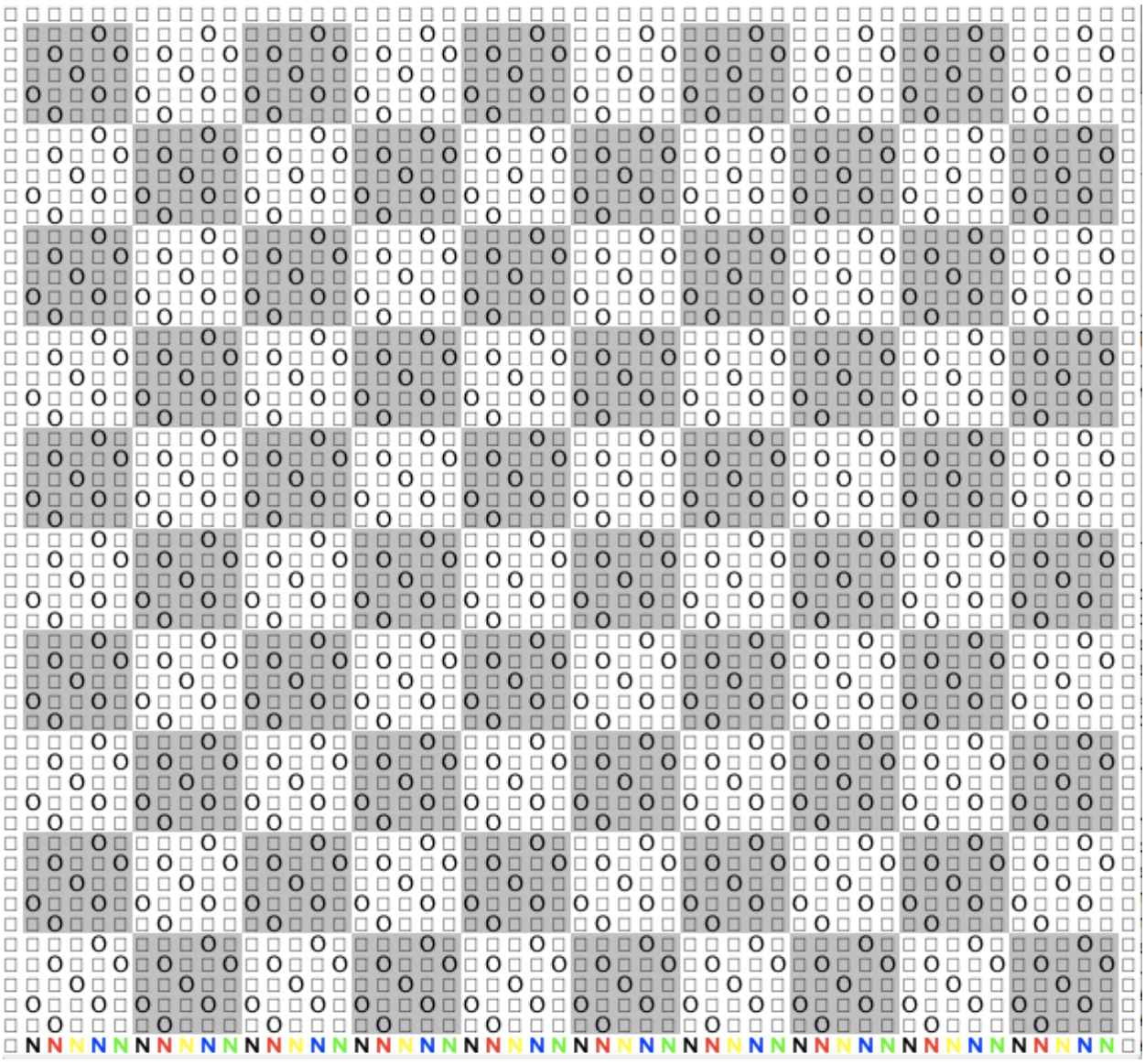

Figure A.14: *World execution "West3 Large" at step 0.*

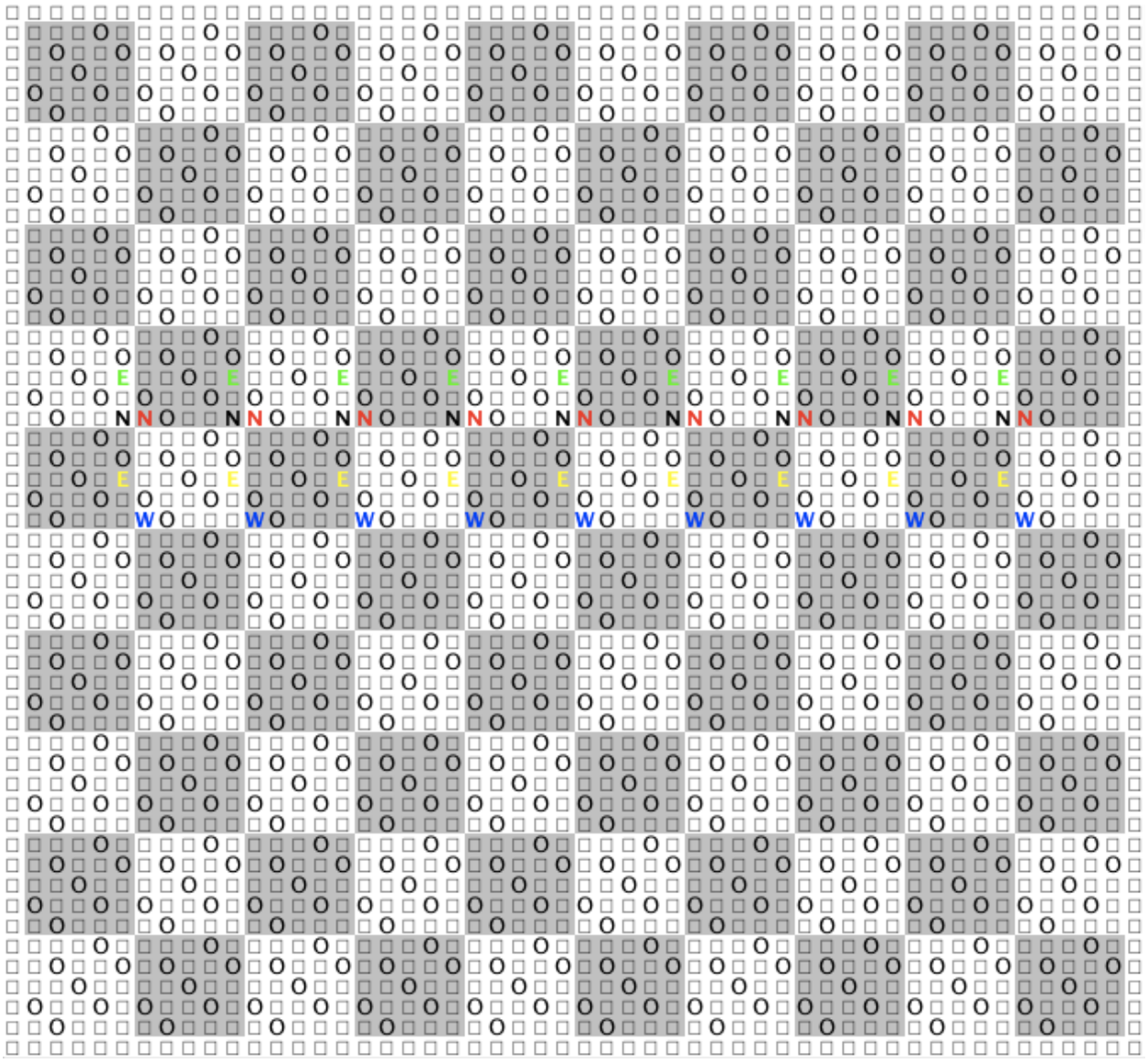

Figure A.15: *World execution "West3 Large" at step 70.*

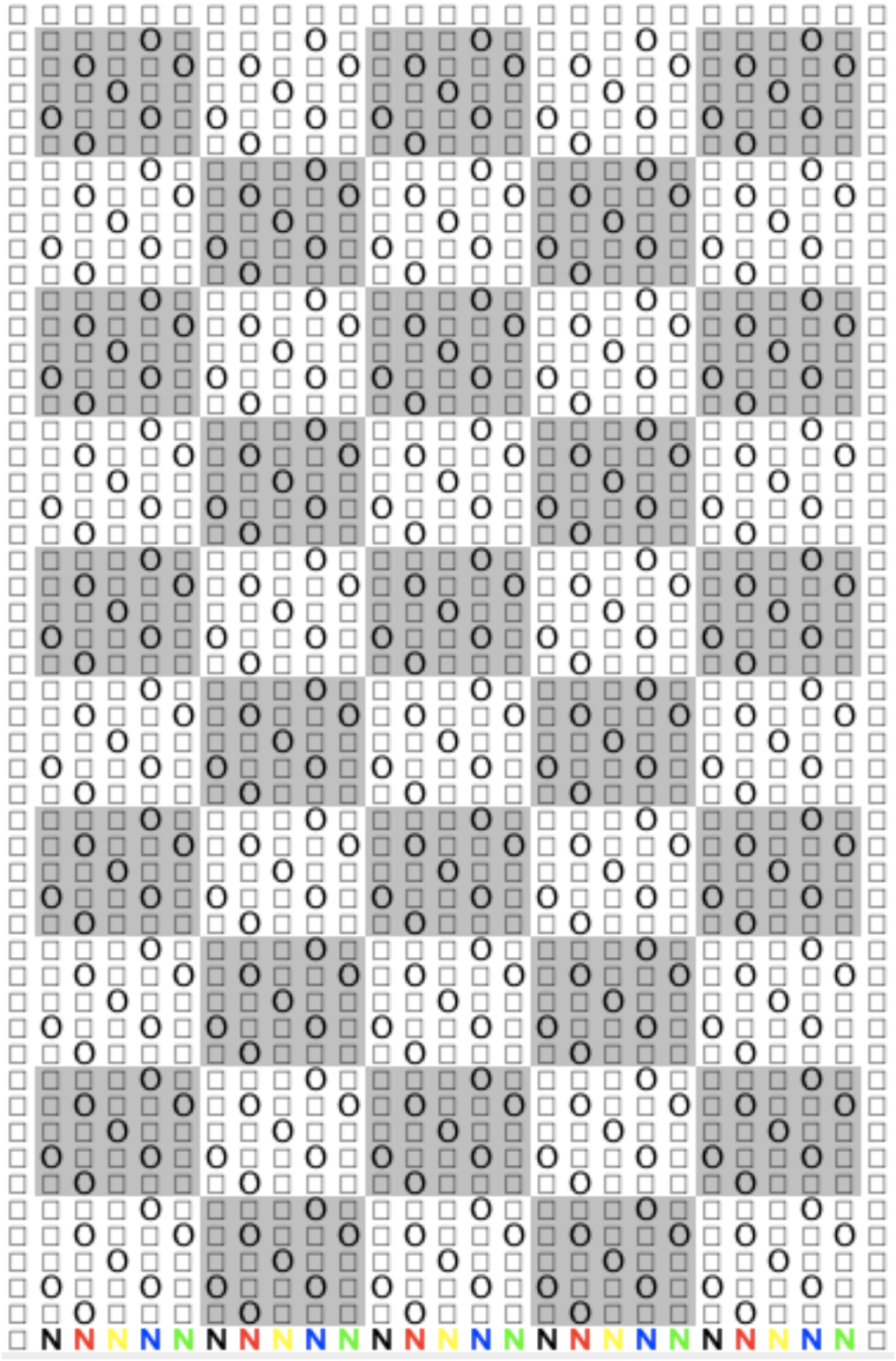

Figure A.16: *World execution "West4" at step 0.*

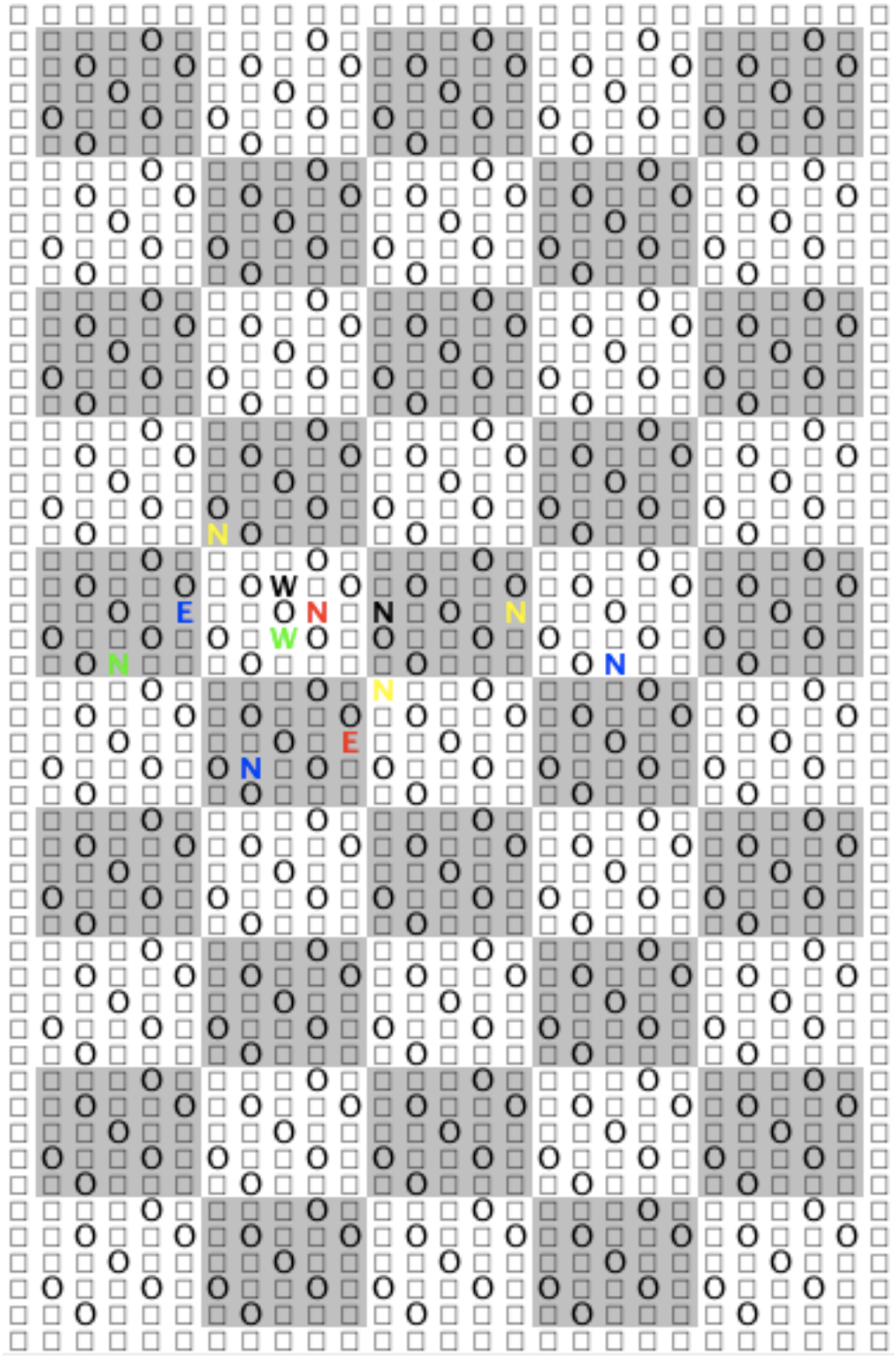

Figure A.17: *World execution "West4" at step 70.*

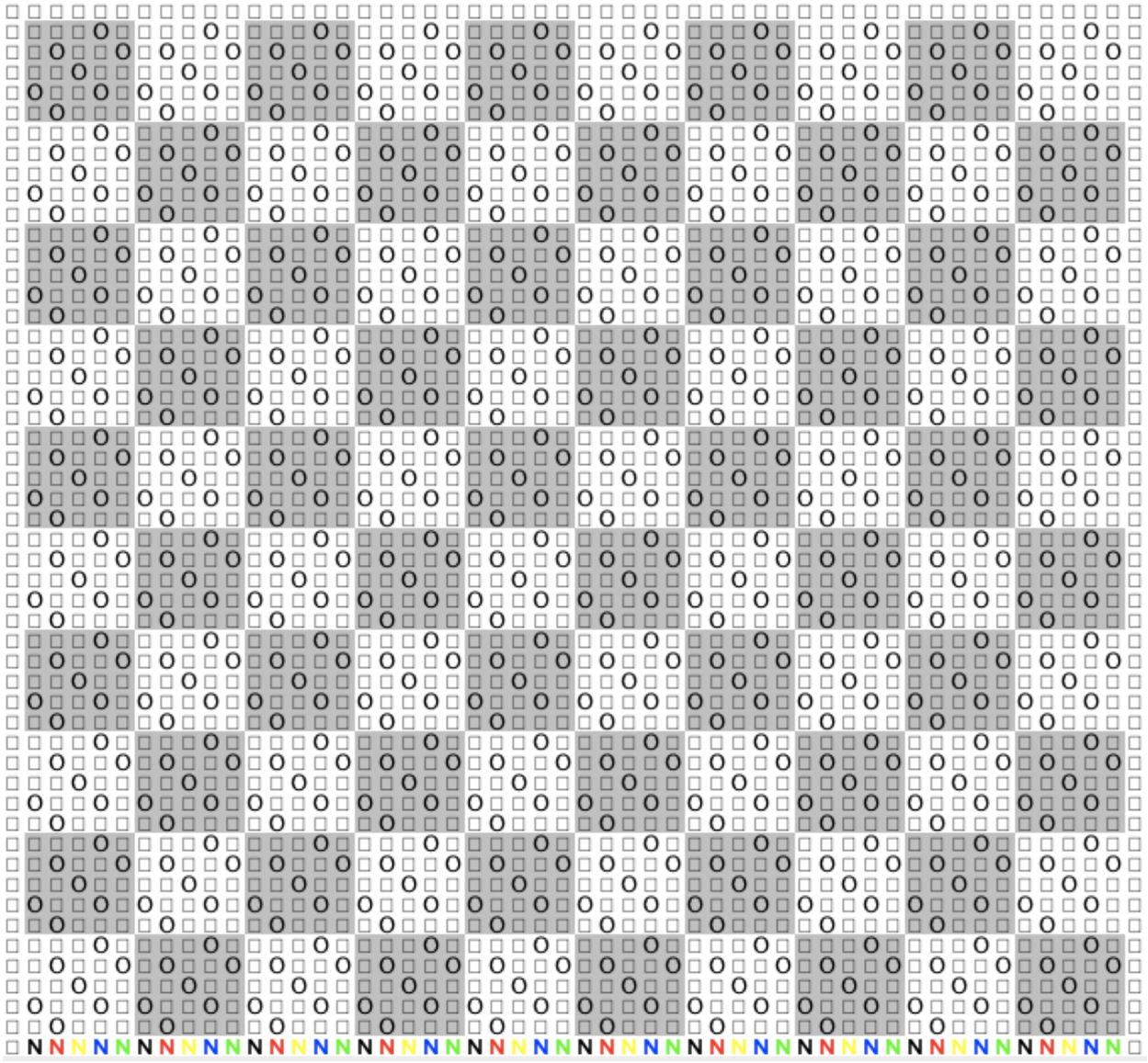

Figure A.18: *World execution "West4 Large" at step 0.*

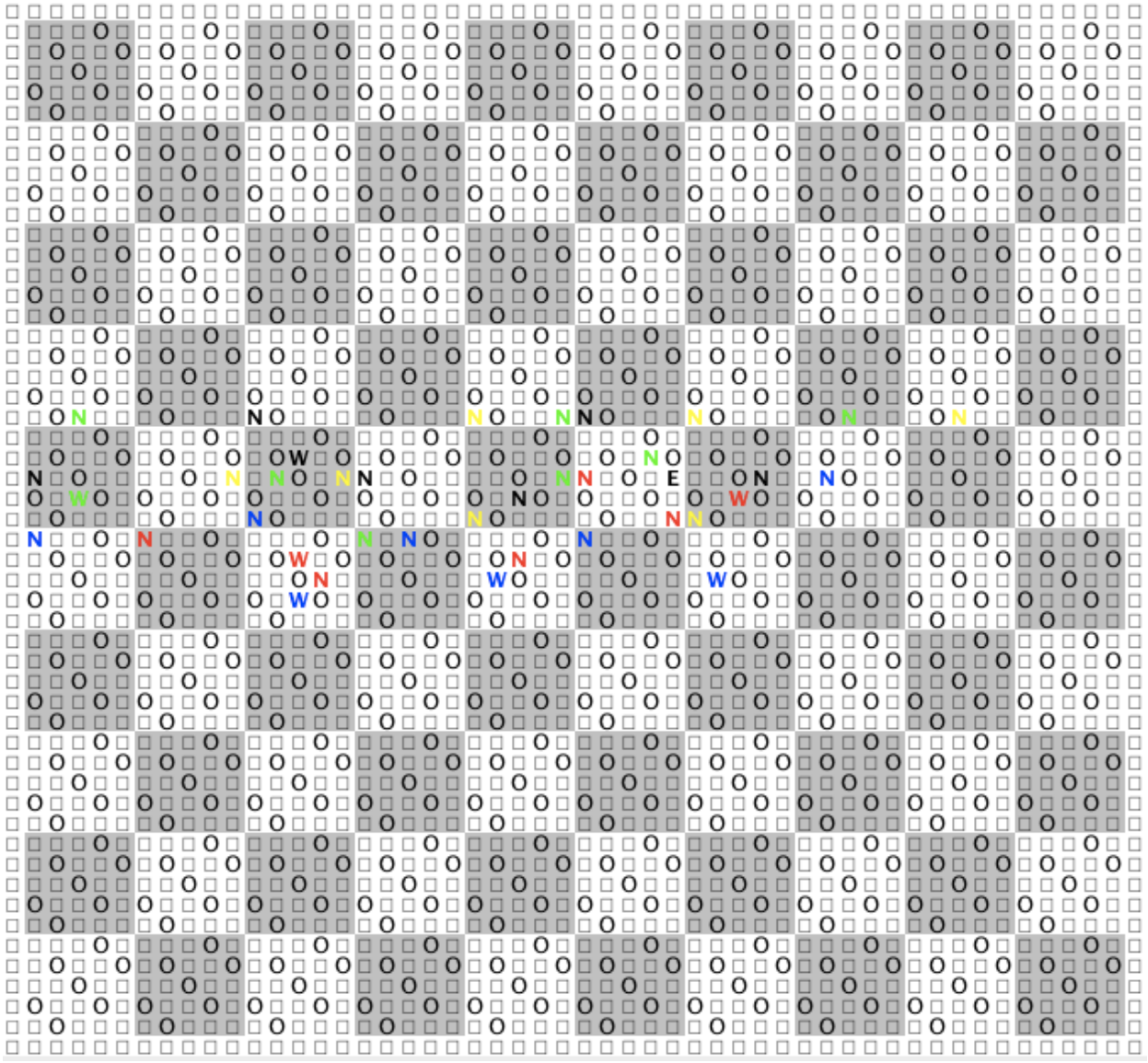

Figure A.19: *World execution "West4 Large" at step 70.*

Appendix B

Simulations

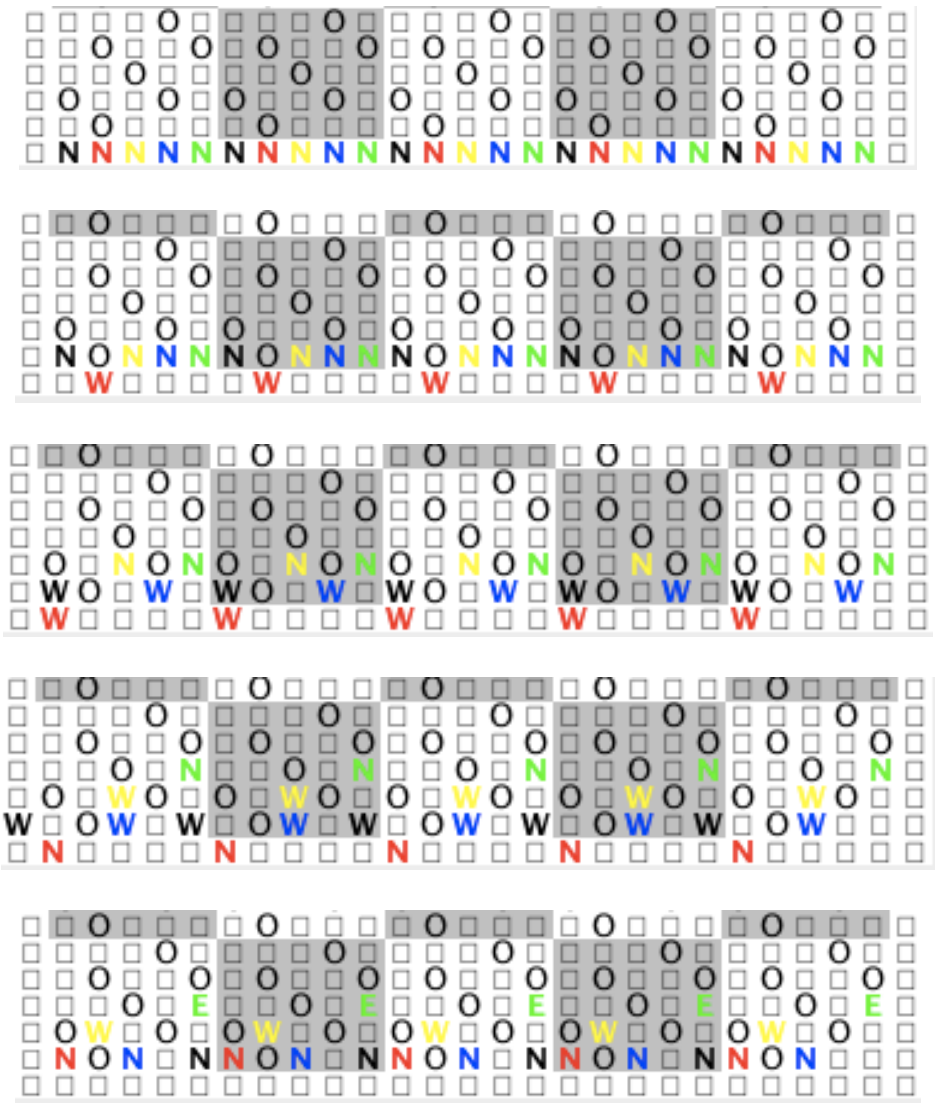

Figure B.1: *Row I of world execution "West3" at steps 0-4.*

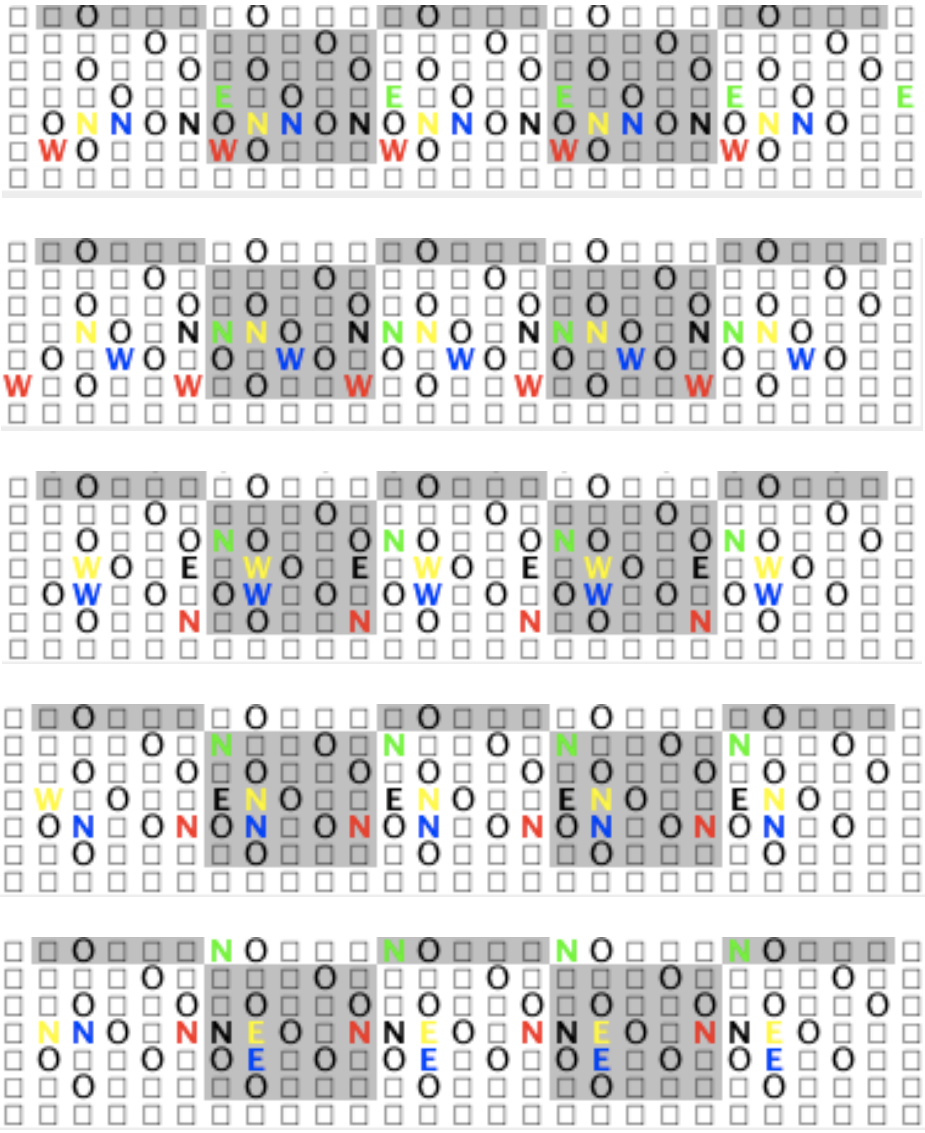

Figure B.2: *Row I of world execution "West3" at steps 5-9.*

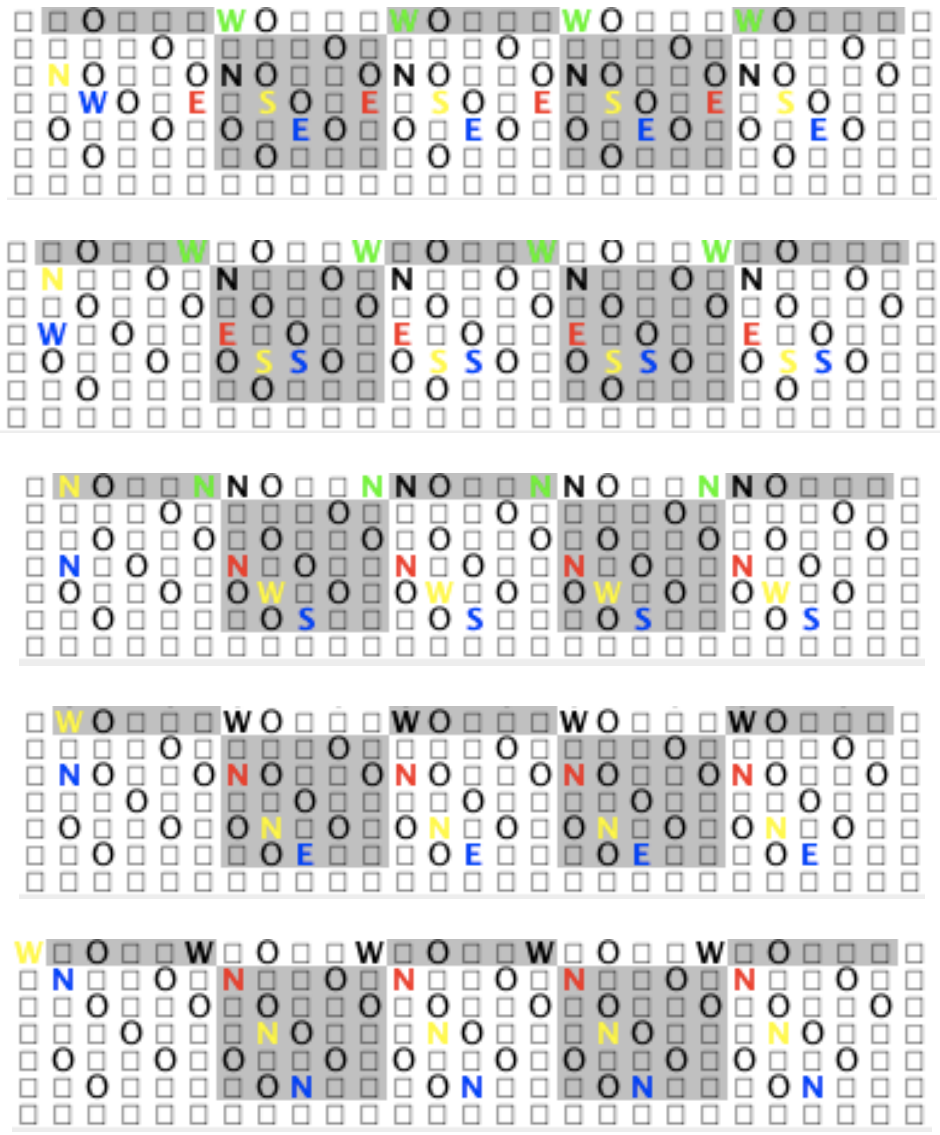

Figure B.3: *Row I of world execution "West3" at steps 10-14.*

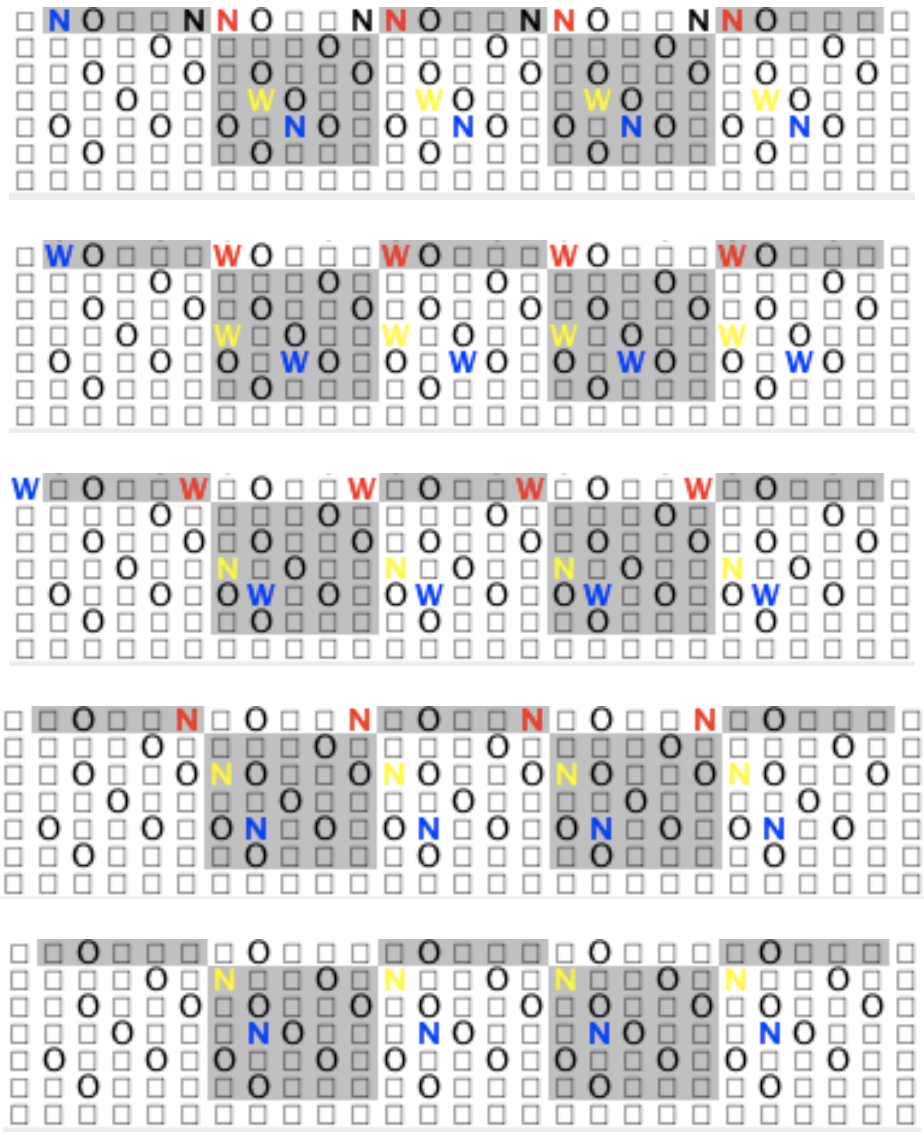

Figure B.4: *Row I of world execution "West3" at steps 15-19.*

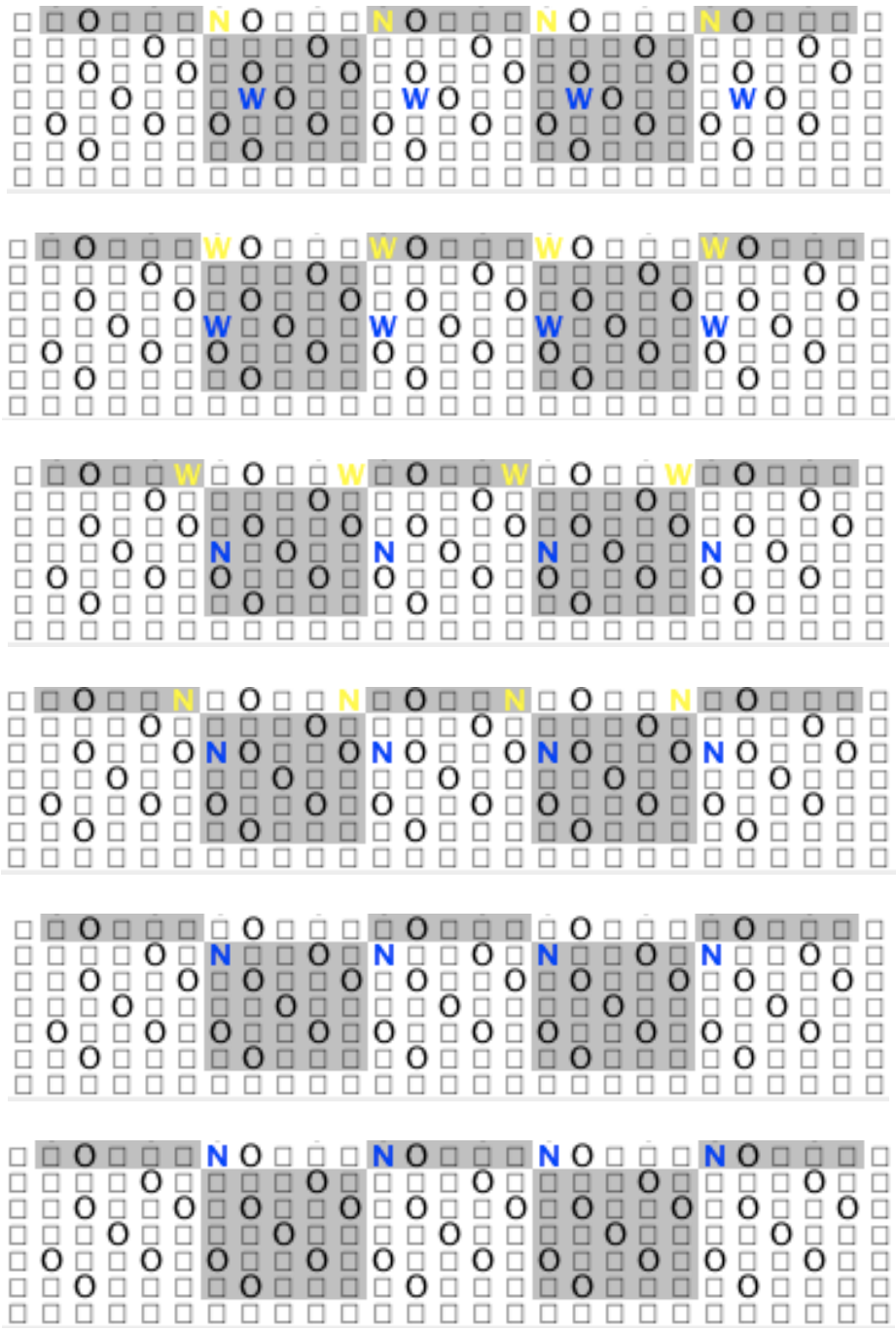

Figure B.5: *Row I of world execution "West3" at steps 20-25.*

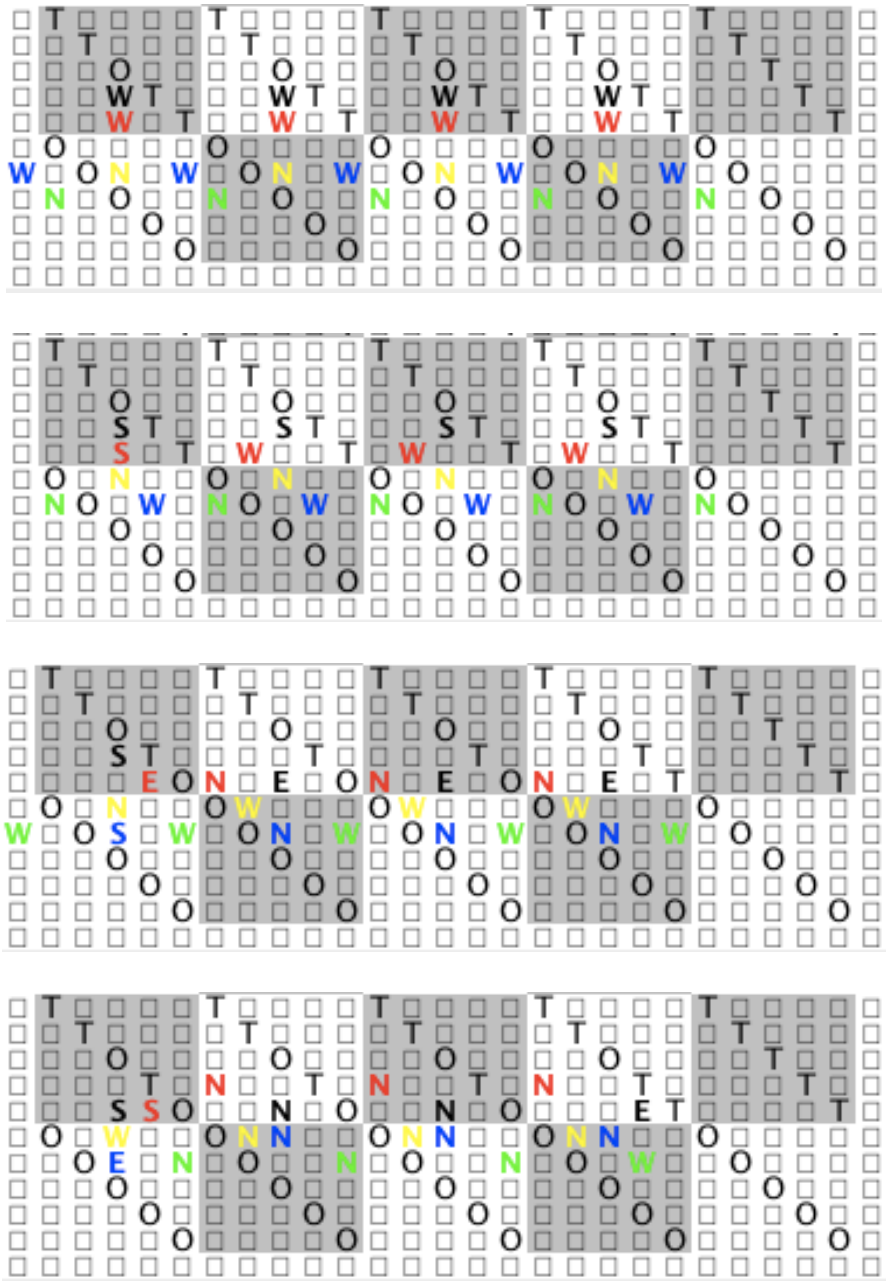

Figure B.6: *World execution "West2" at steps 15 and 16, 18, and 19.*

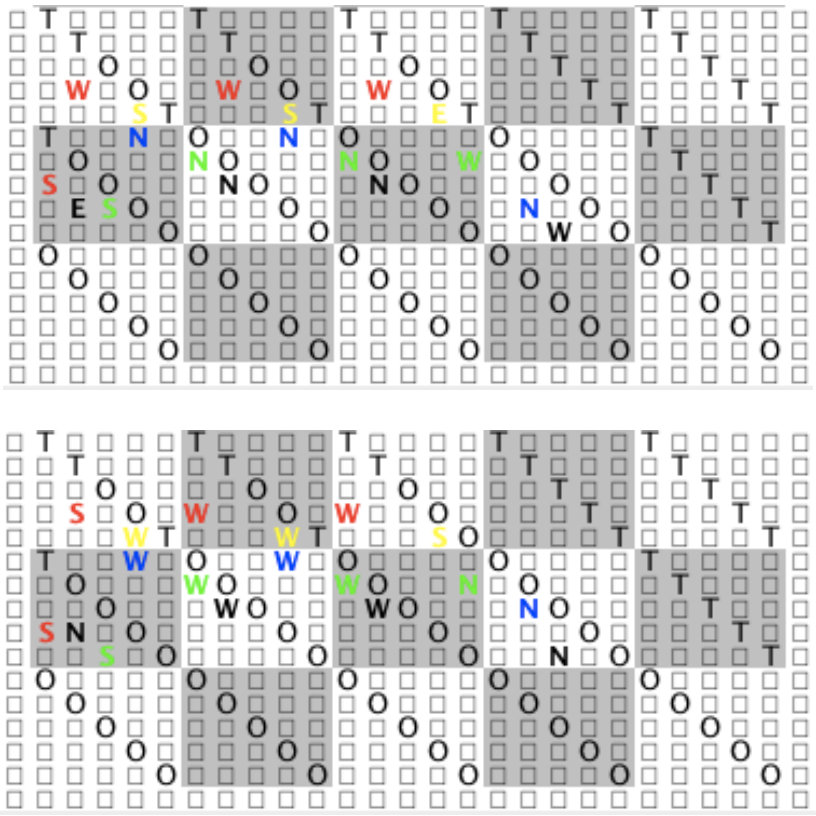

Figure B.7: *World execution "West2" at steps 34 and 35.*

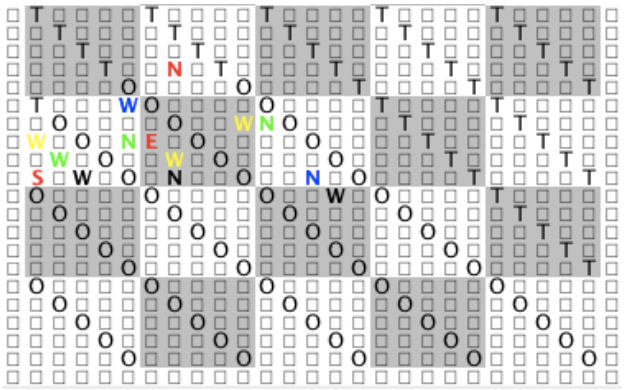

Figure B.8: *World execution "West2" at step 50.*

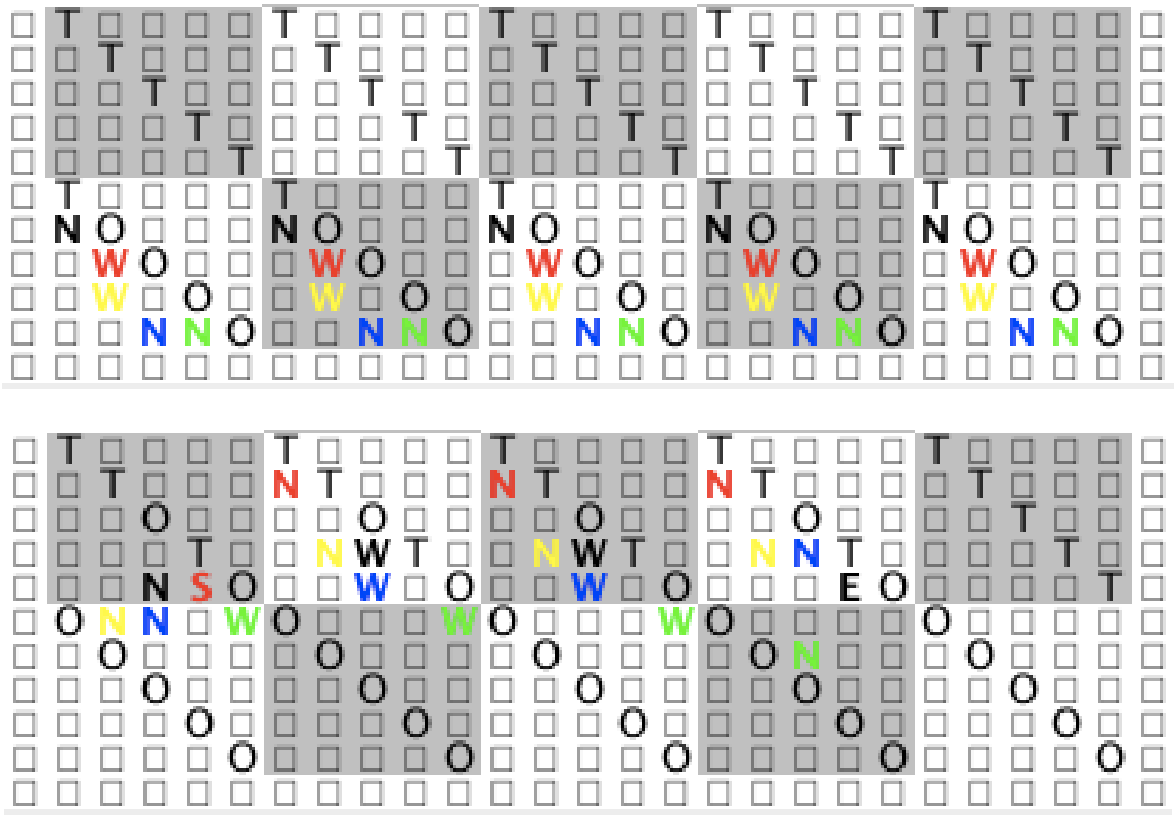

Figure B.9: *World execution "West2" at steps 4 and 21.*

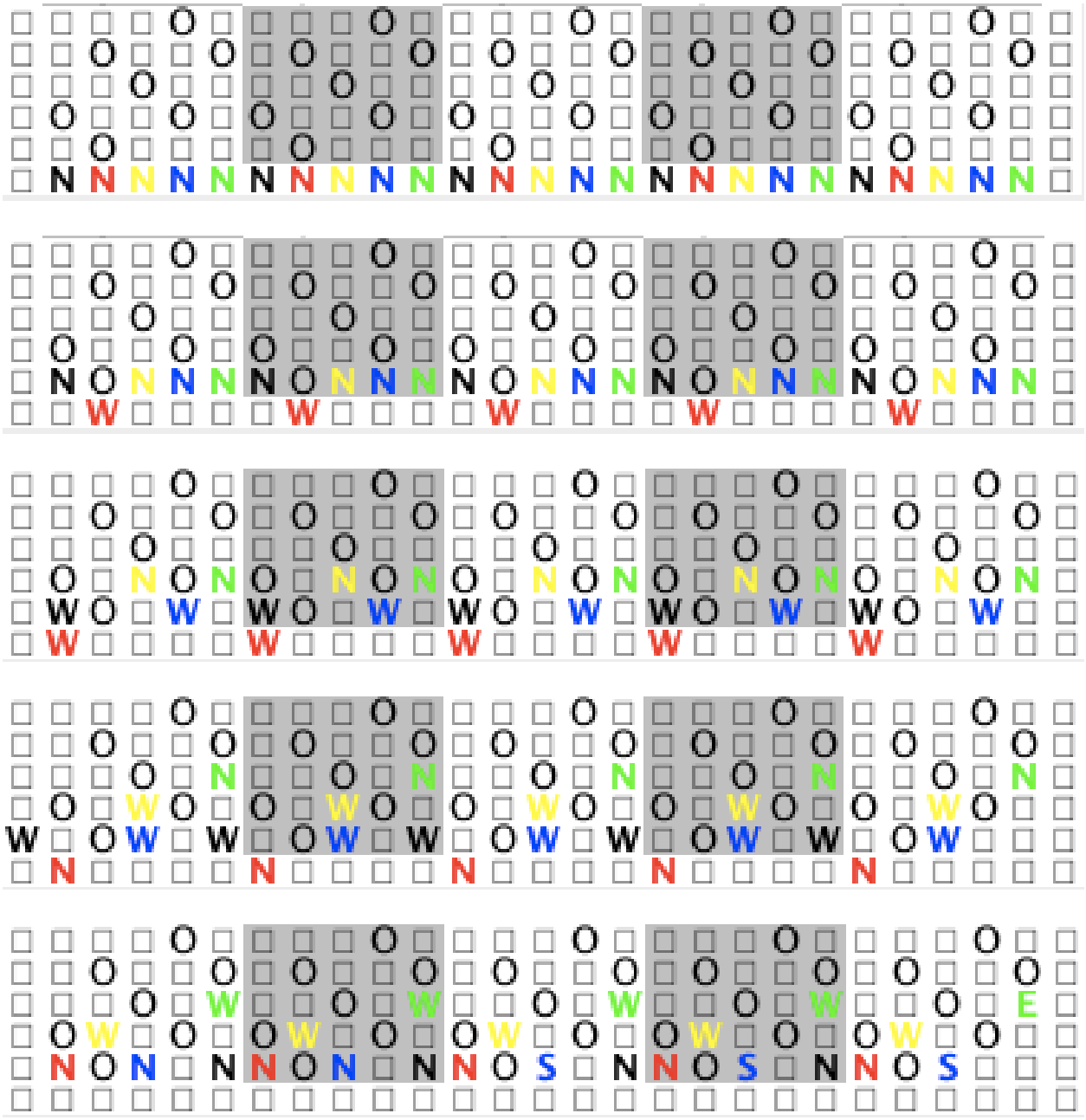

Figure B.10: *World execution "West4" at steps 0-4.*

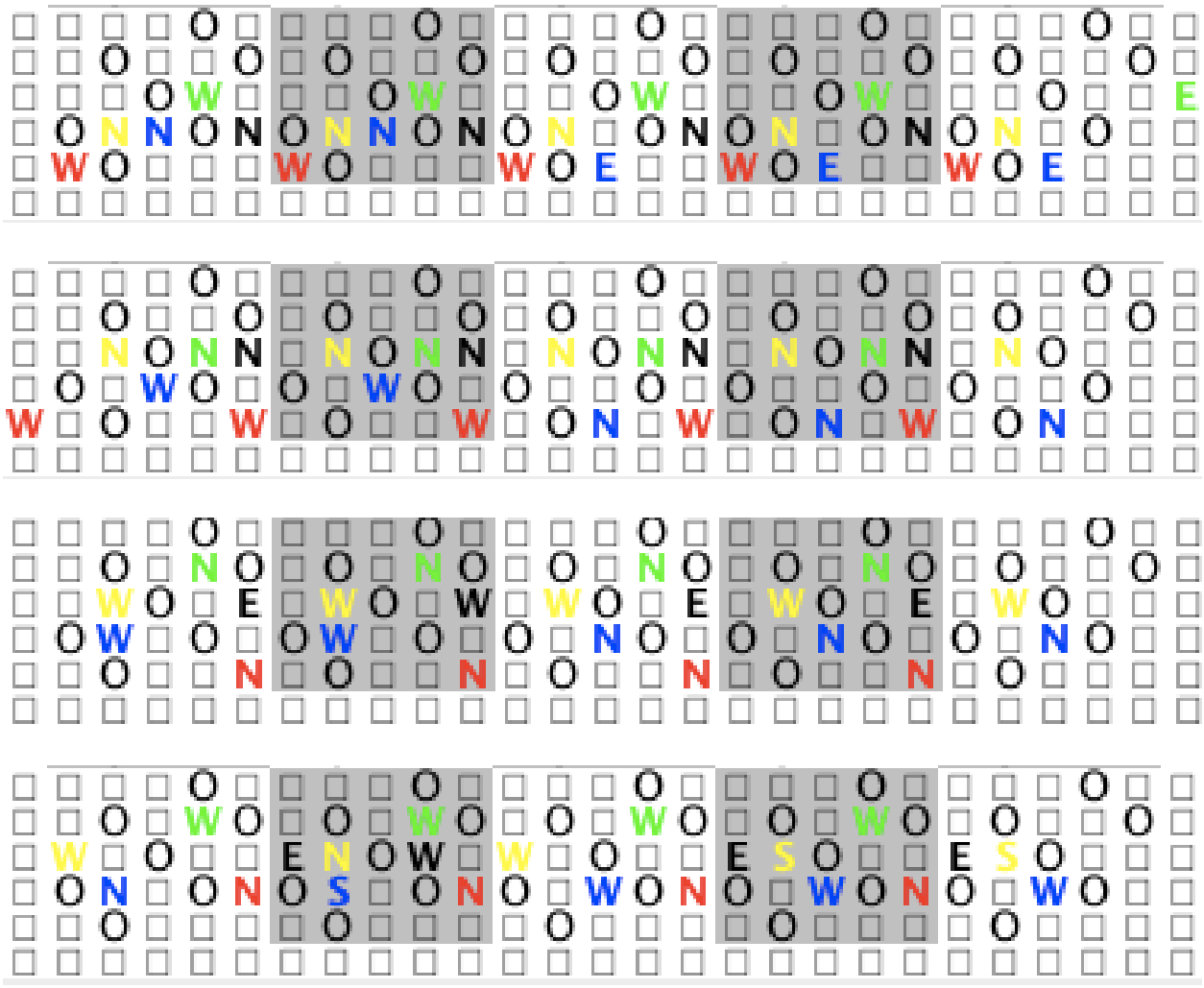

Figure B.11: *World execution "West4" at steps 5-8.*

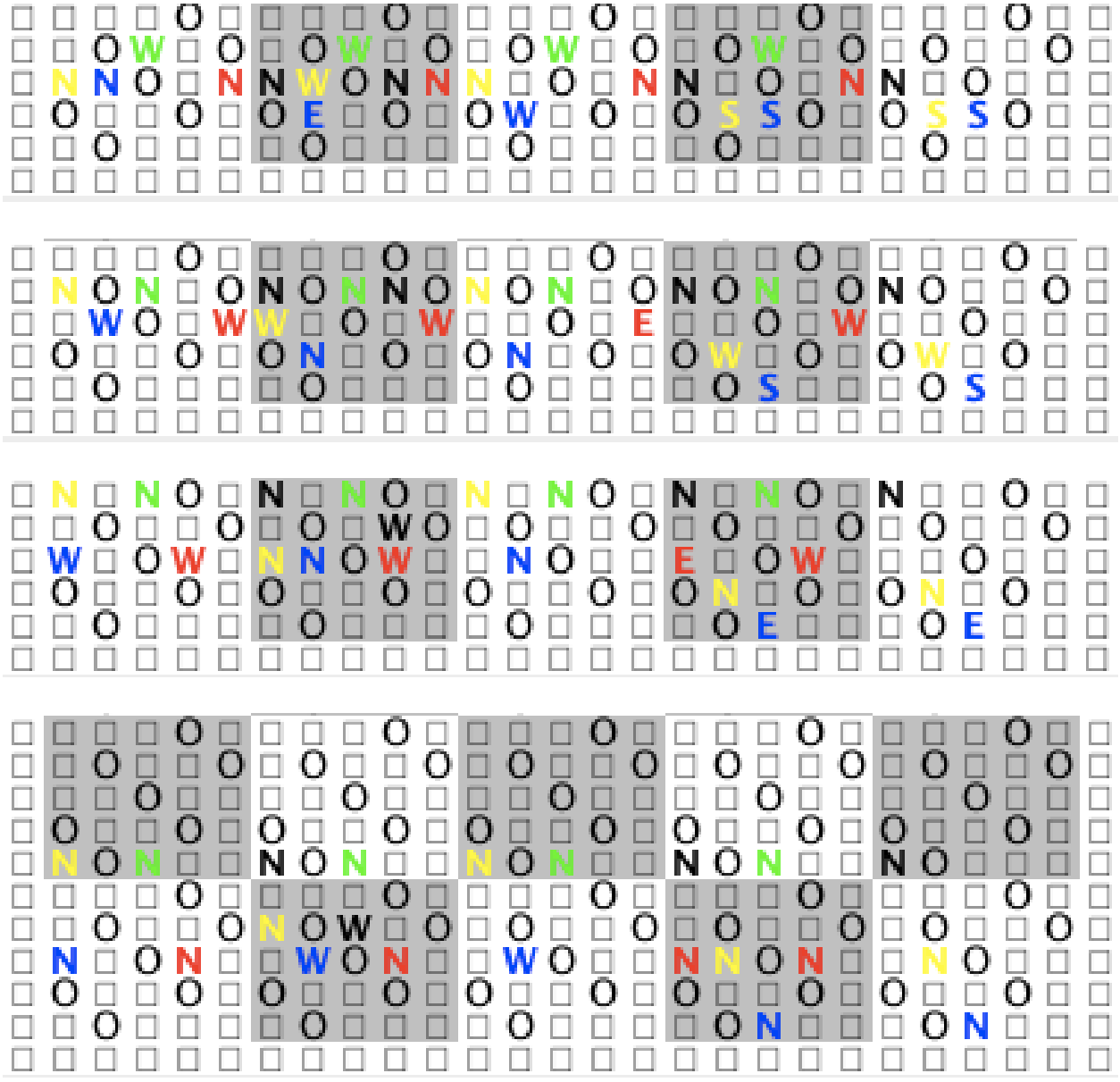

Figure B.12: *World execution "West4" at steps 9-12.*

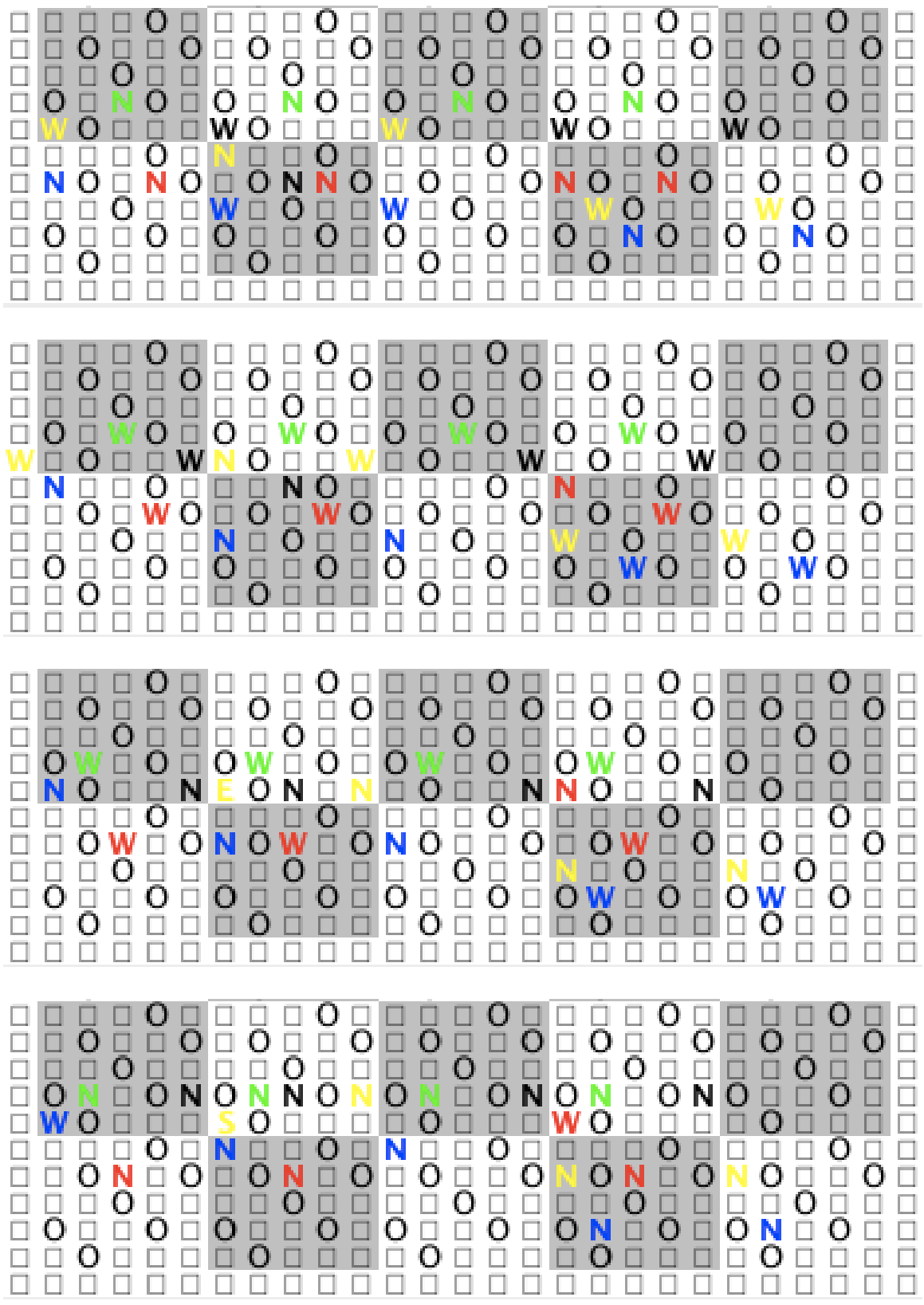

Figure B.13: *World execution "West4" at steps 13-16.*

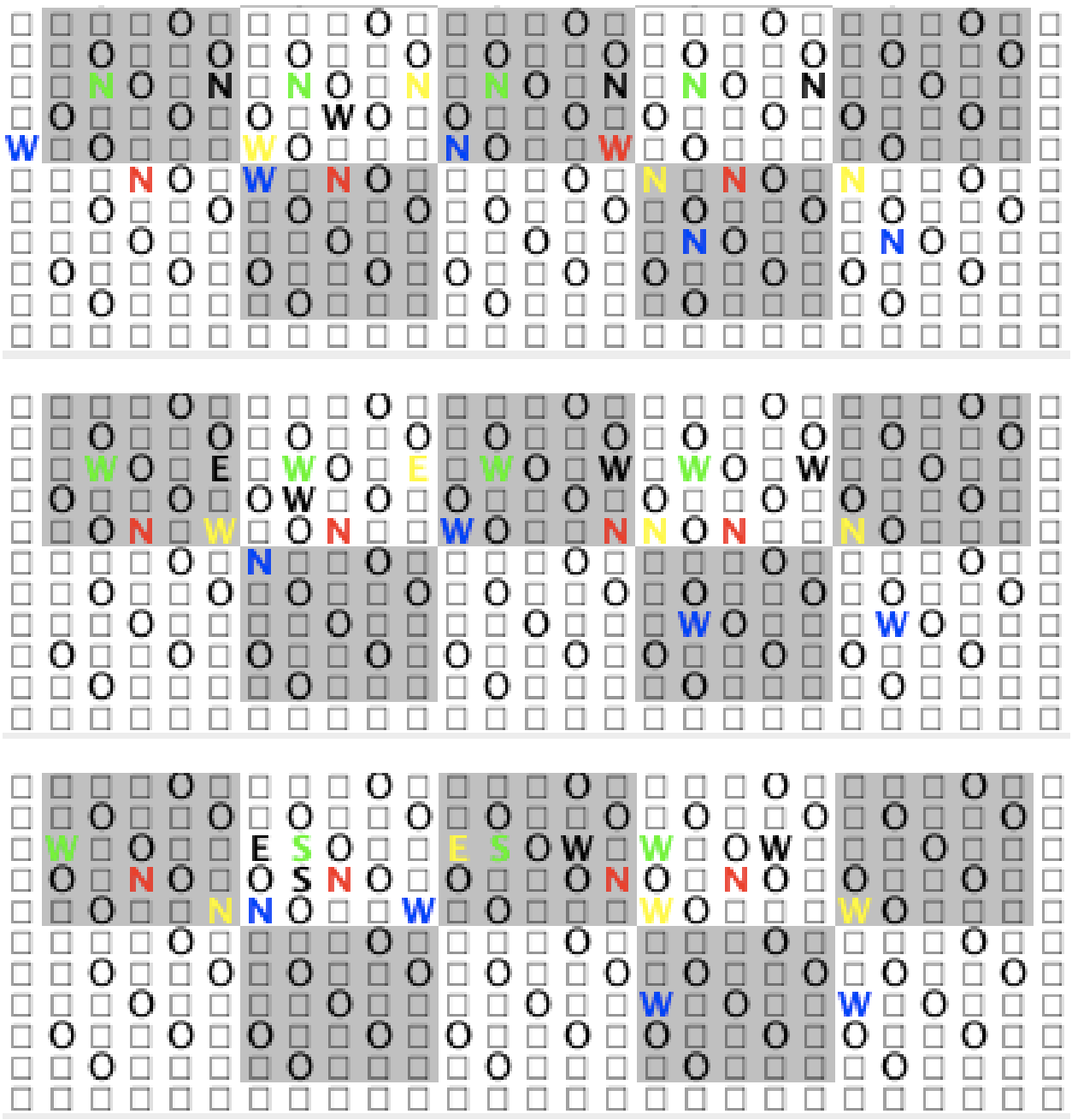

Figure B.14: *World execution "West4" at steps 17-19.*

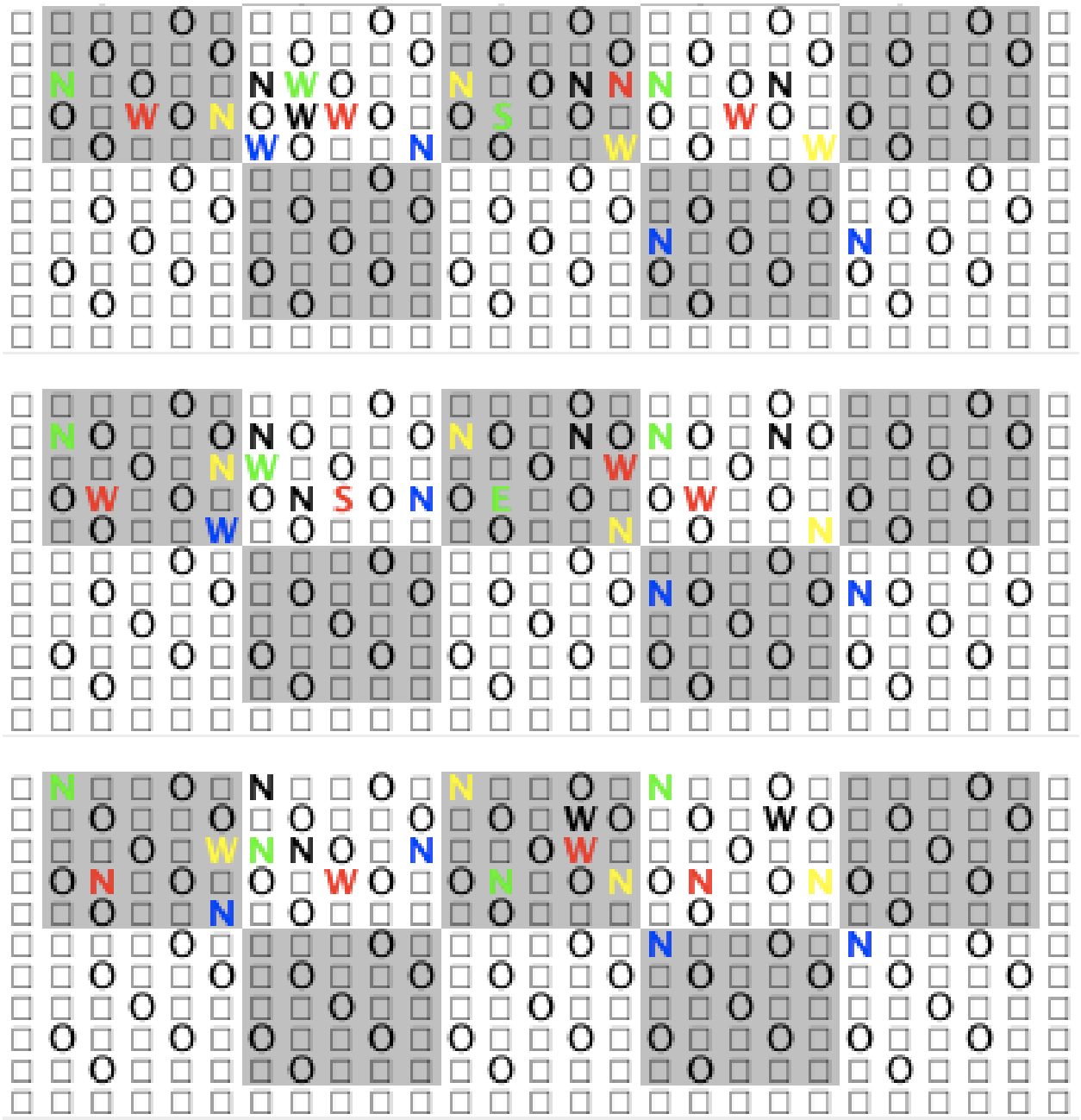

Figure B.15: *World execution "West4" at steps 20-22.*

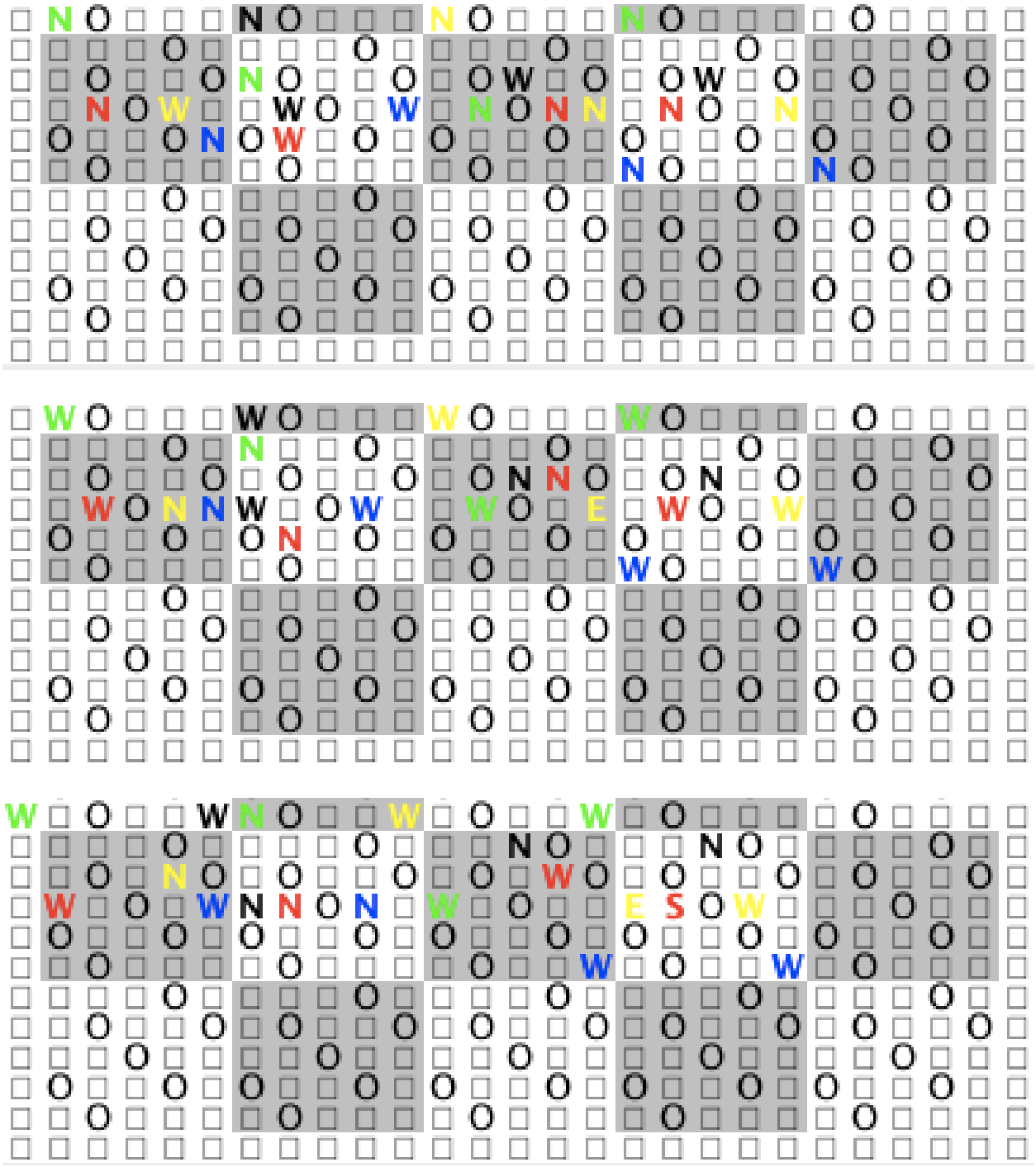

Figure B.16: *World execution "West4" at steps 23-25.*

Appendix C

Supplemental Predictive Value Visuals

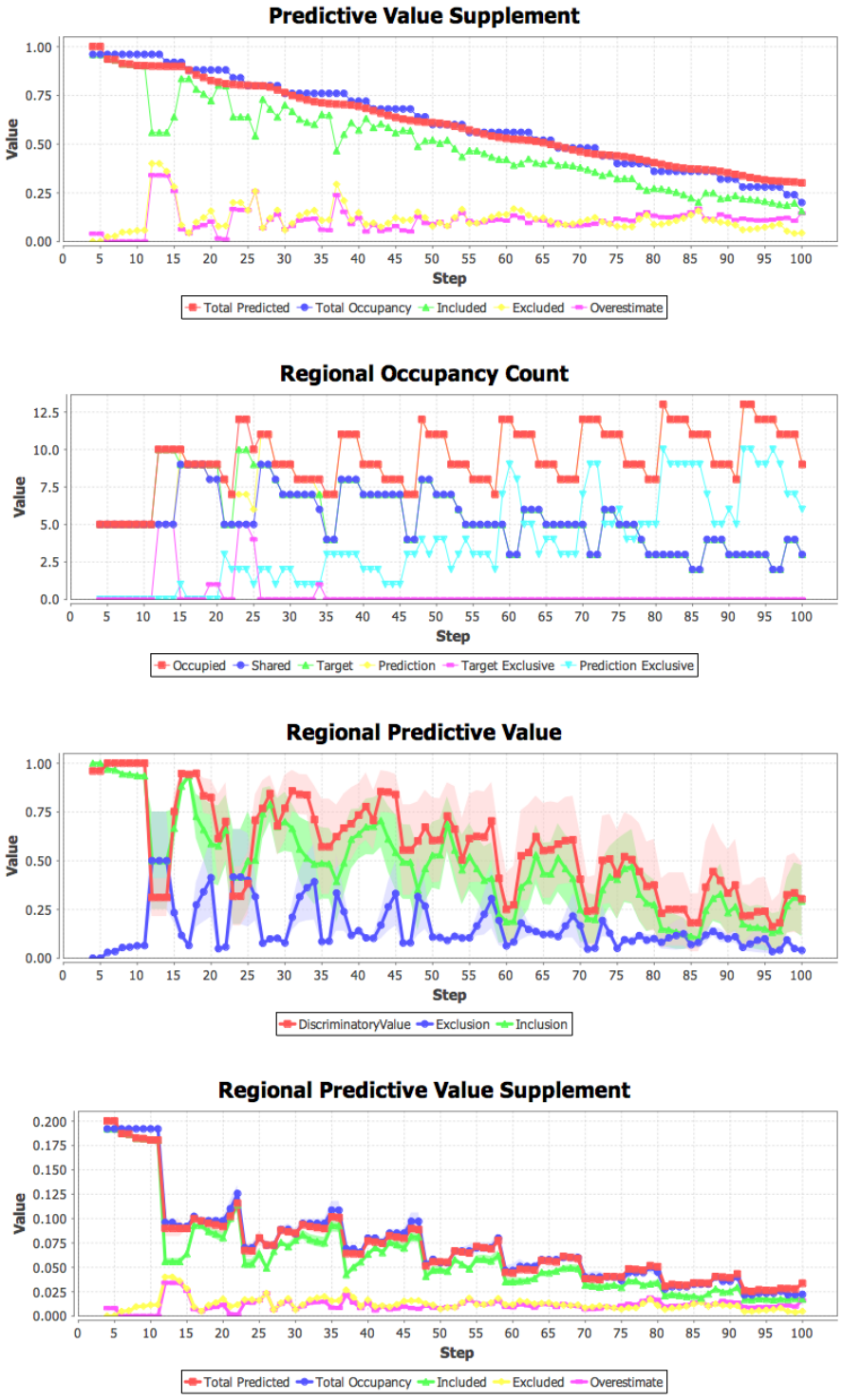

Figure C.1: *Additional predictive value plots for graph "East" and batch execution "East".*

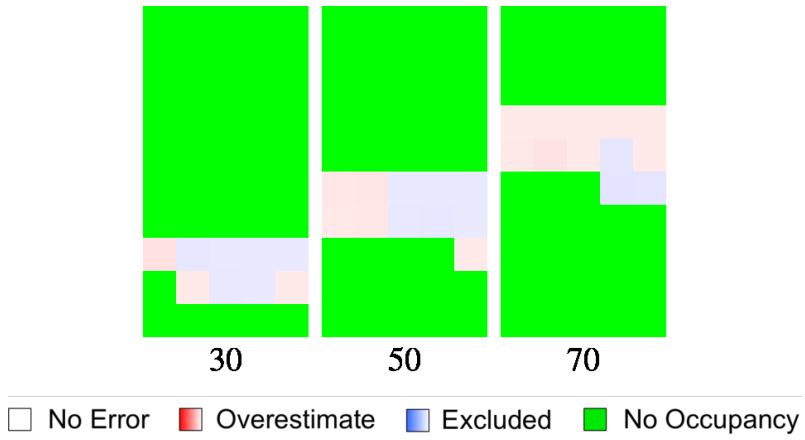

Figure C.2: *Excluded and overestimate heatmaps for graph "East" and batch execution "East".*

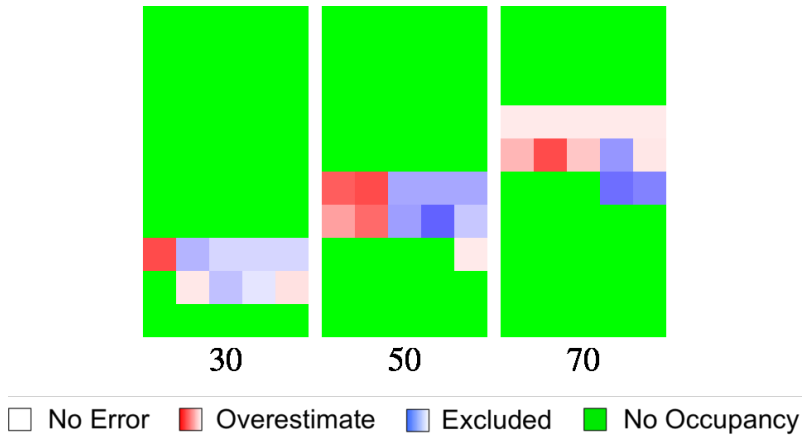

Figure C.3: *Normalized excluded and overestimate heatmaps for graph "East" and batch execution "East".*

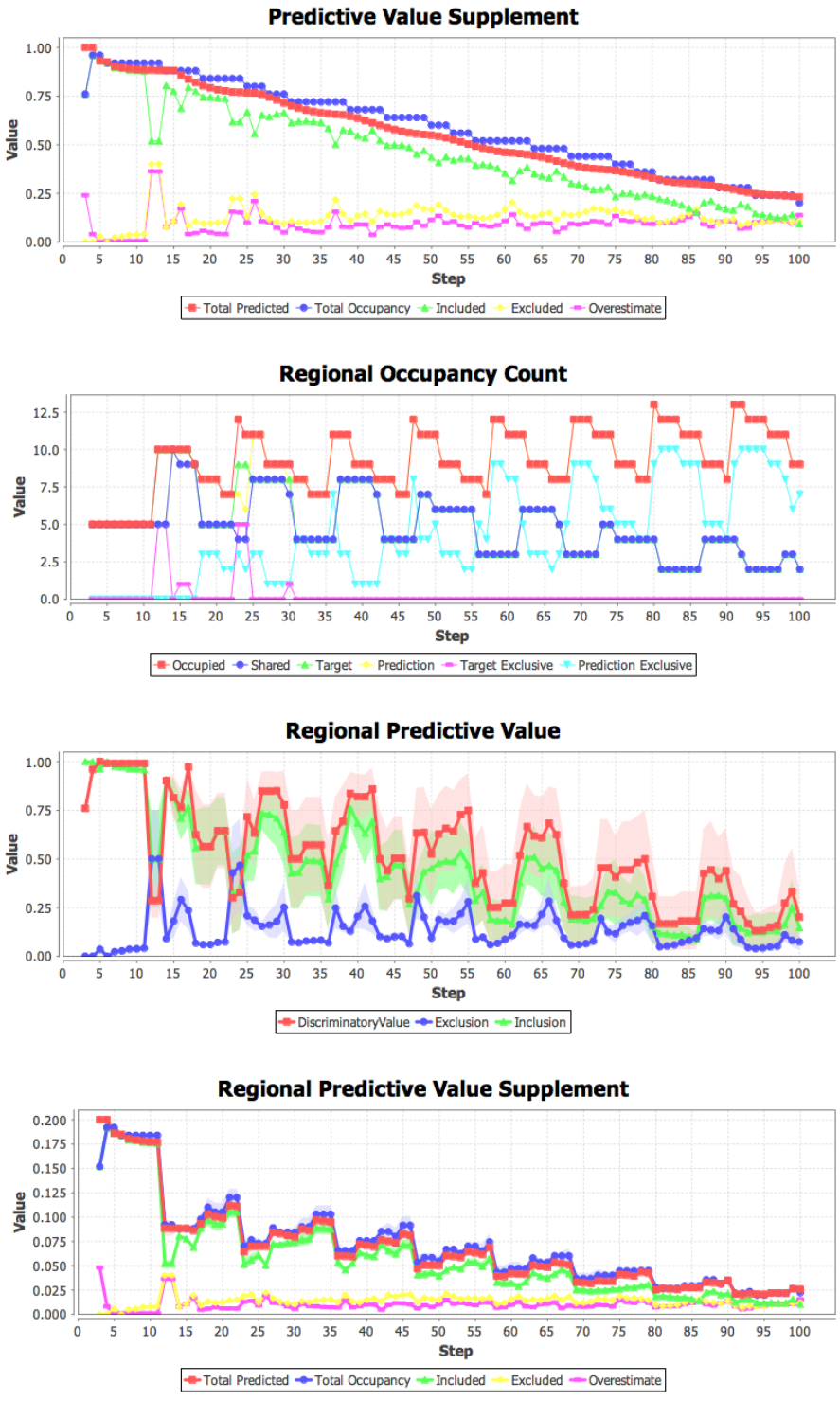

Figure C.4: *Additional predictive value plots for graph "West1 A" and batch execution "West1".*

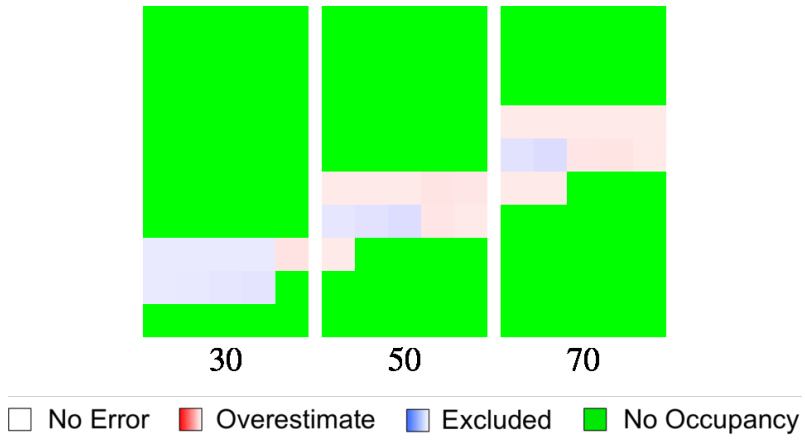

Figure C.5: *Excluded and overestimate heatmaps for graph "West1 A" and batch execution "West1".*

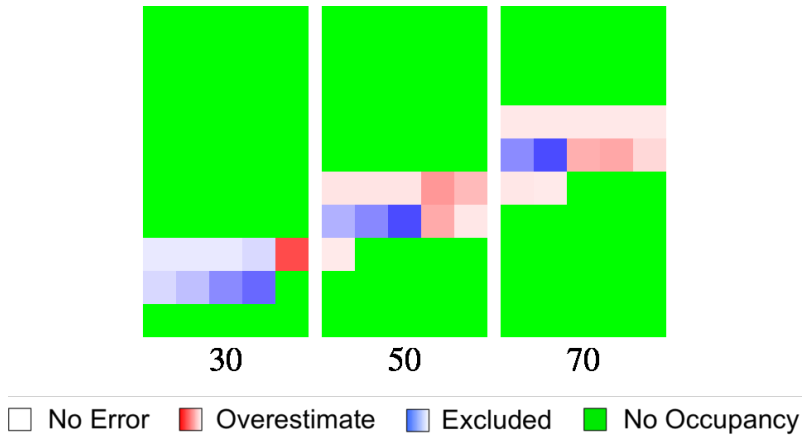

Figure C.6: *Normalized excluded and overestimate heatmaps for graph "West1 A" and batch execution "West1".*

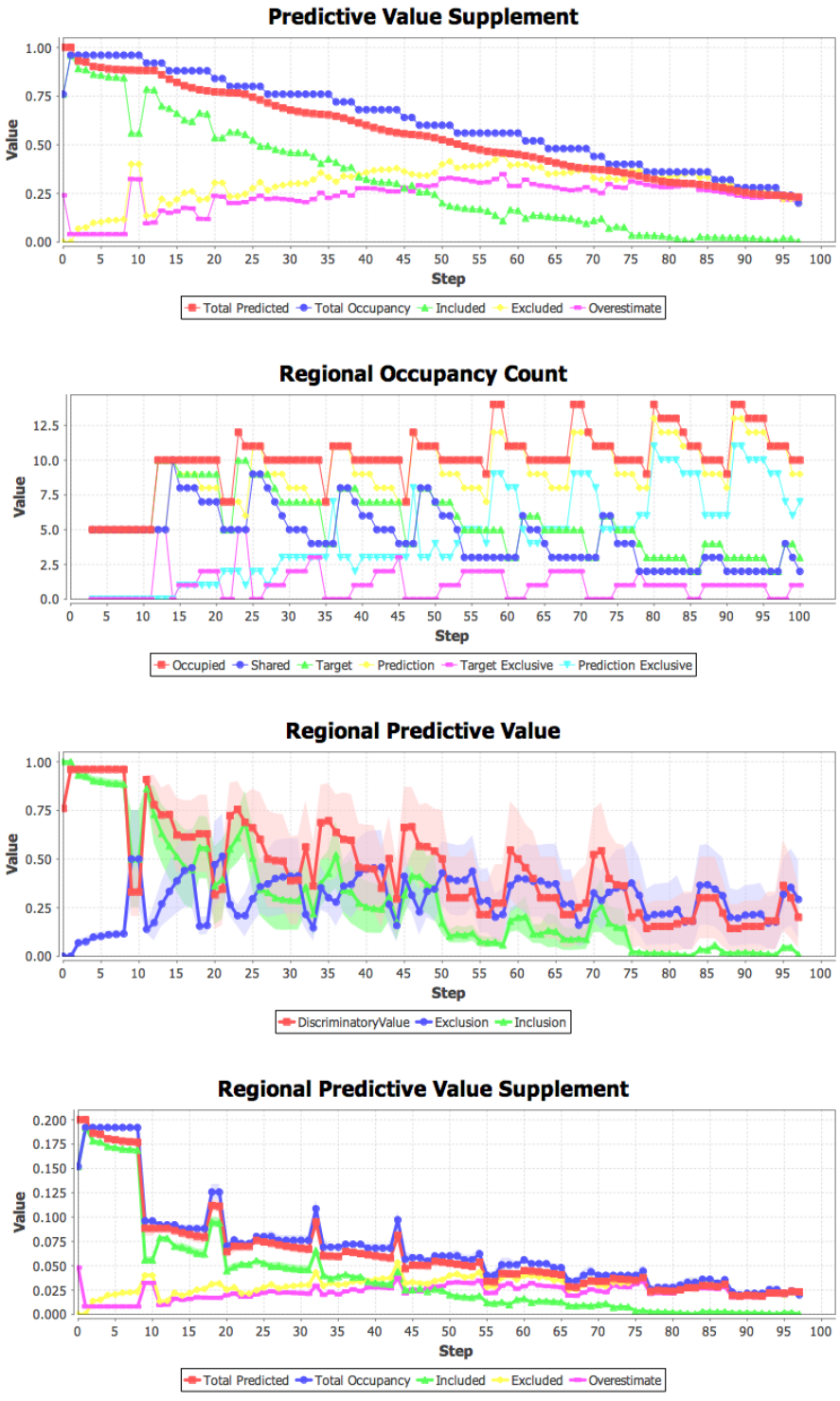

Figure C.7: *Additional predictive value plots for graph "West1 A" and batch execution "East".*

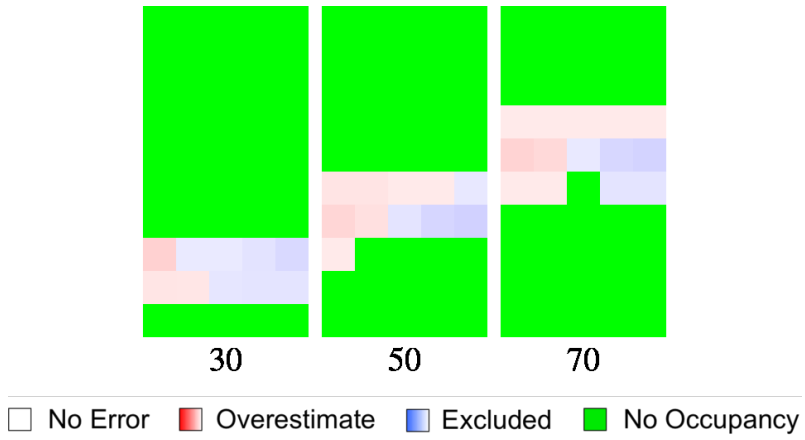

Figure C.8: *Excluded and overestimate heatmaps for graph "West1 A" and batch execution "East".*

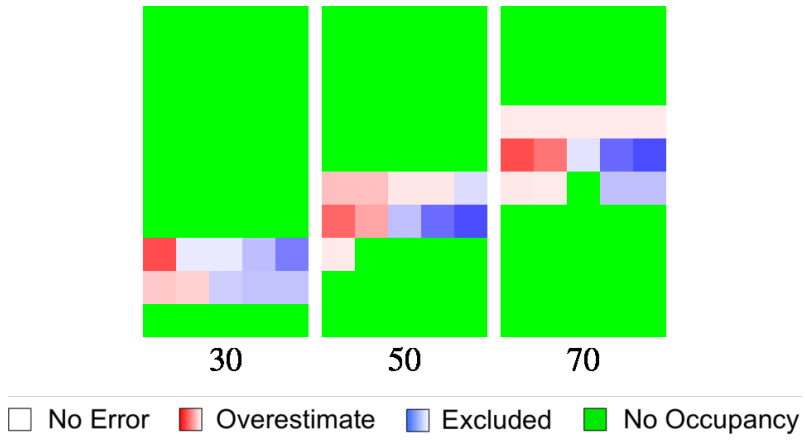

Figure C.9: *Normalized excluded and overestimate heatmaps for graph "West1 A" and batch execution "East".*

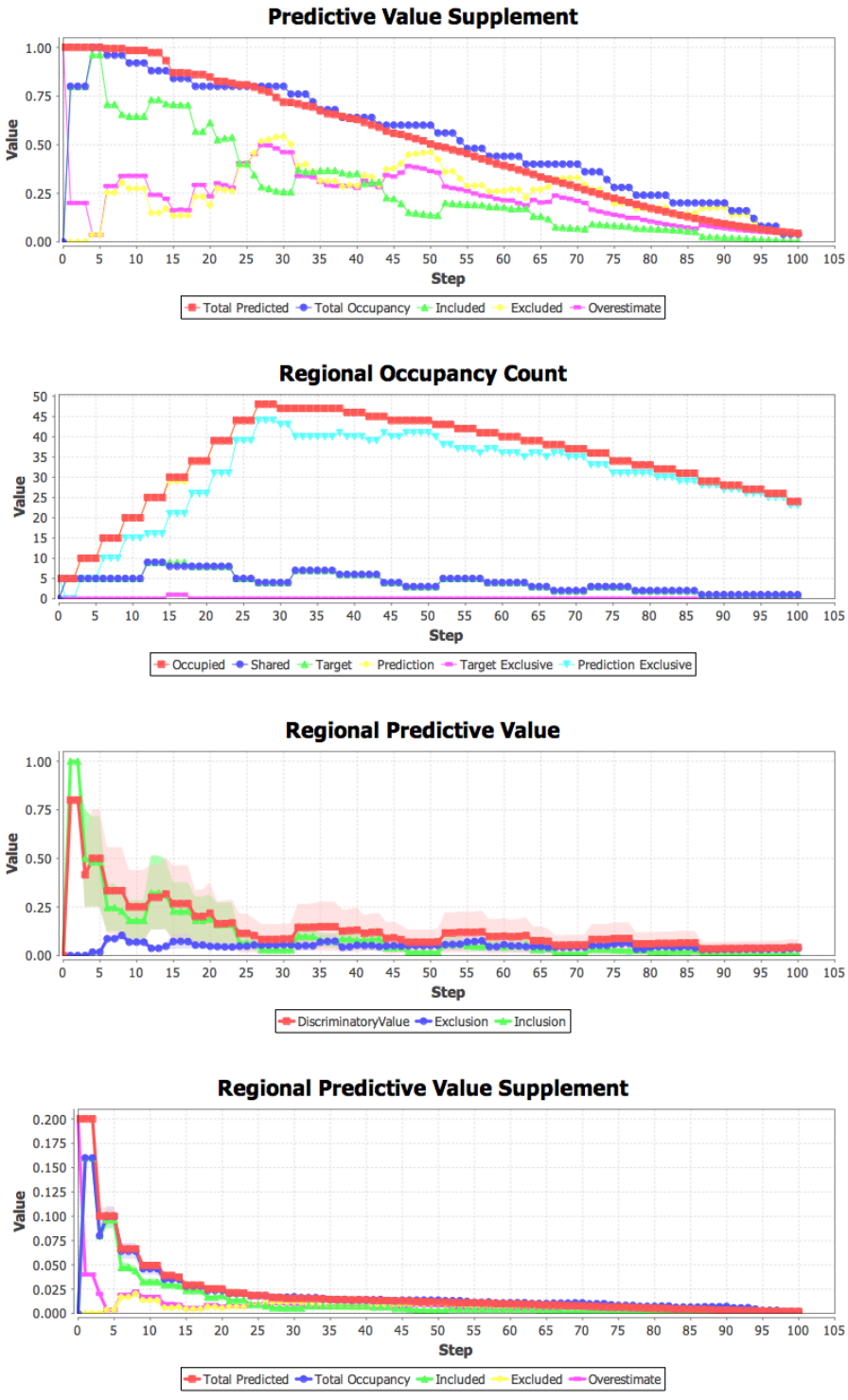

Figure C.10: *Additional predictive value plots for graph "West2 A" and batch execution "West2".*

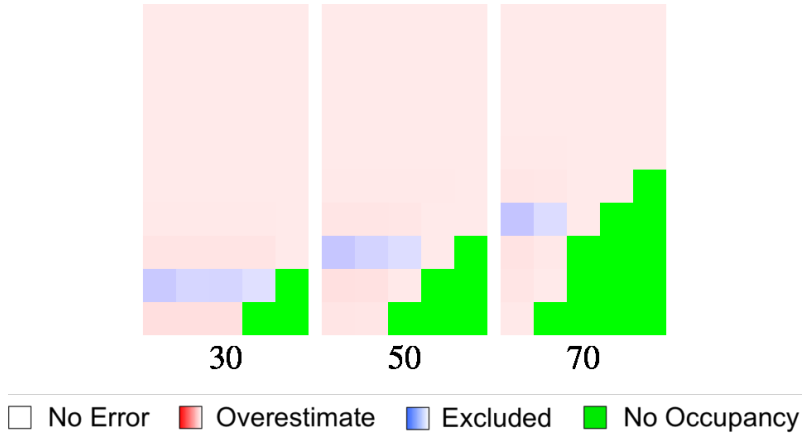

Figure C.11: *Excluded and overestimate heatmaps for graph "West2 A" and batch execution "West2".*

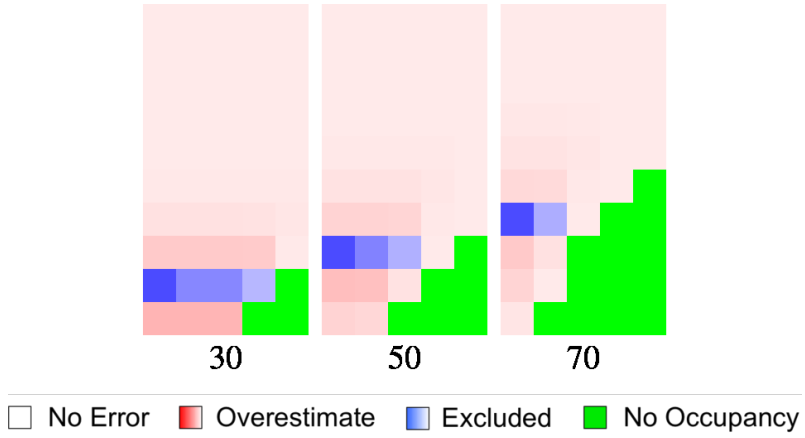

Figure C.12: *Normalized excluded and overestimate heatmaps for "West2 A" and batch execution "West2".*

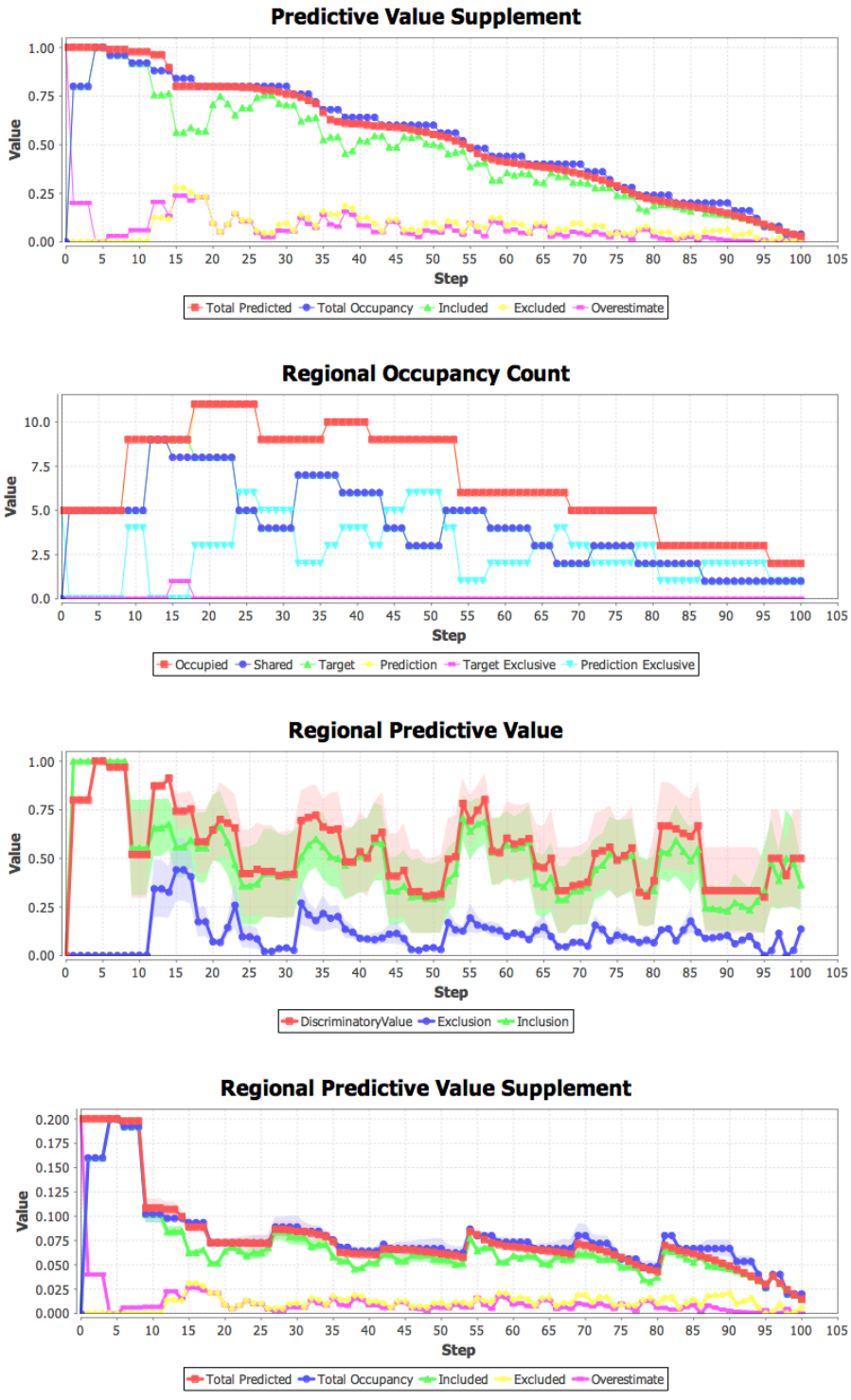

Figure C.13: *Additional predictive value plots for graph "West2 B" and batch execution "West2".*

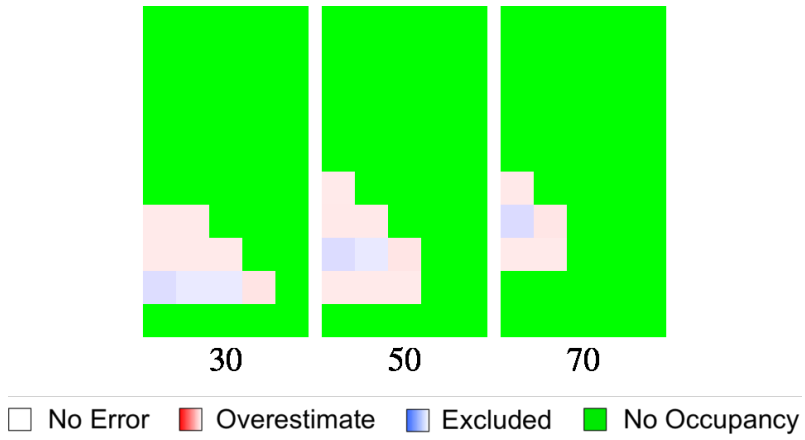

Figure C.14: *Excluded and overestimate heatmaps for graph "West2 B" and batch execution "West2".*

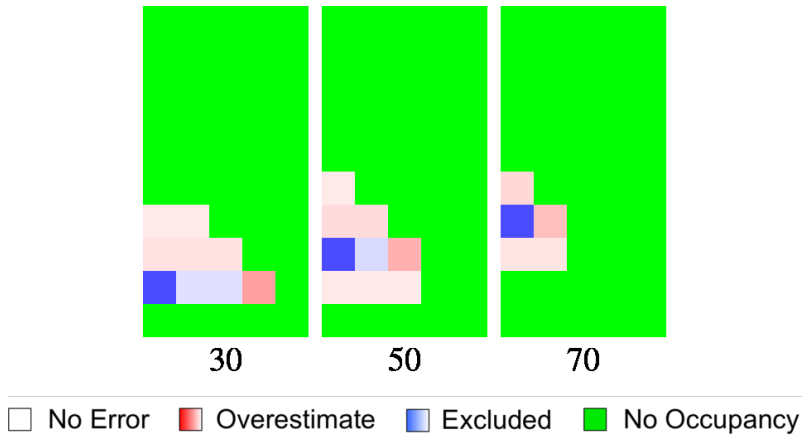

Figure C.15: *Normalized excluded and overestimate heatmaps for "West2 B" and batch execution "West2".*

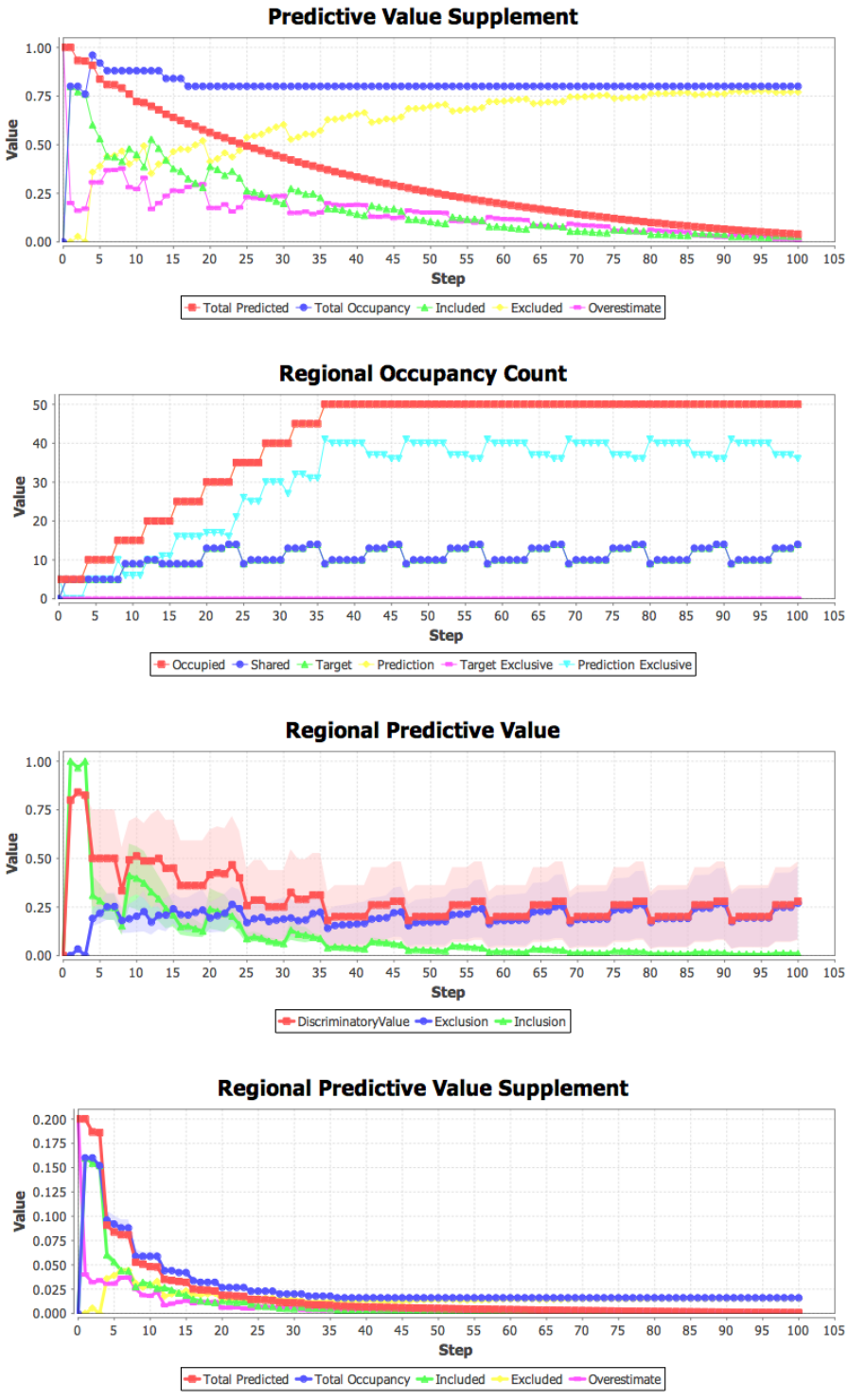

Figure C.16: *Additional predictive value plots for graph "West3 A" and batch execution "West3".*

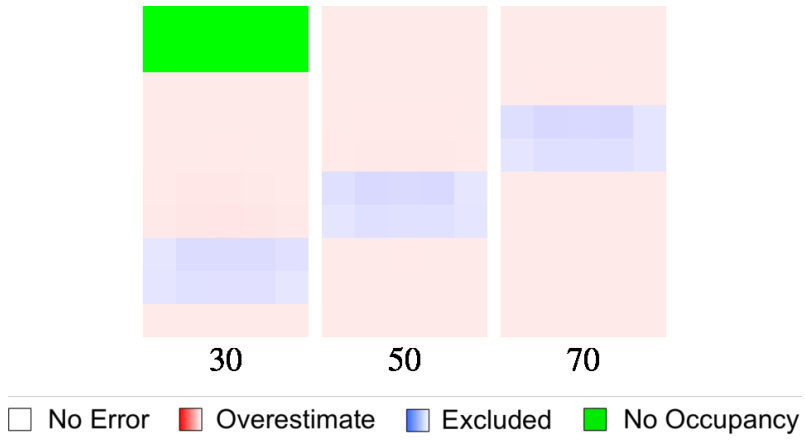

Figure C.17: *Excluded and overestimate heatmaps for graph "West3 A" and batch execution "West3".*

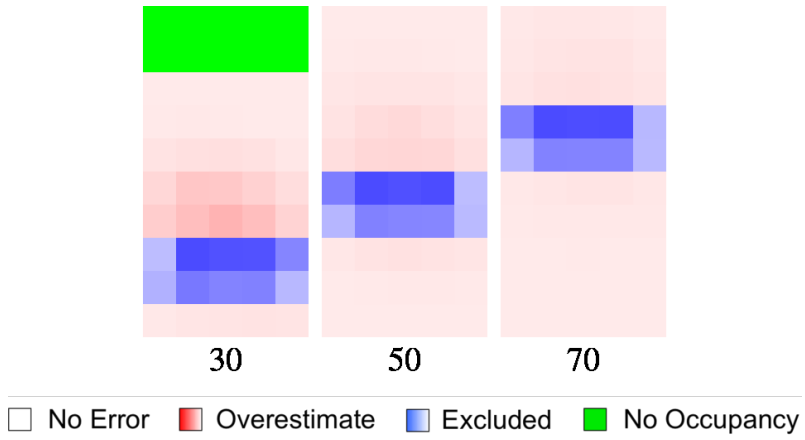

Figure C.18: *Normalized excluded and overestimate heatmaps for "West3 A" and batch execution "West3".*

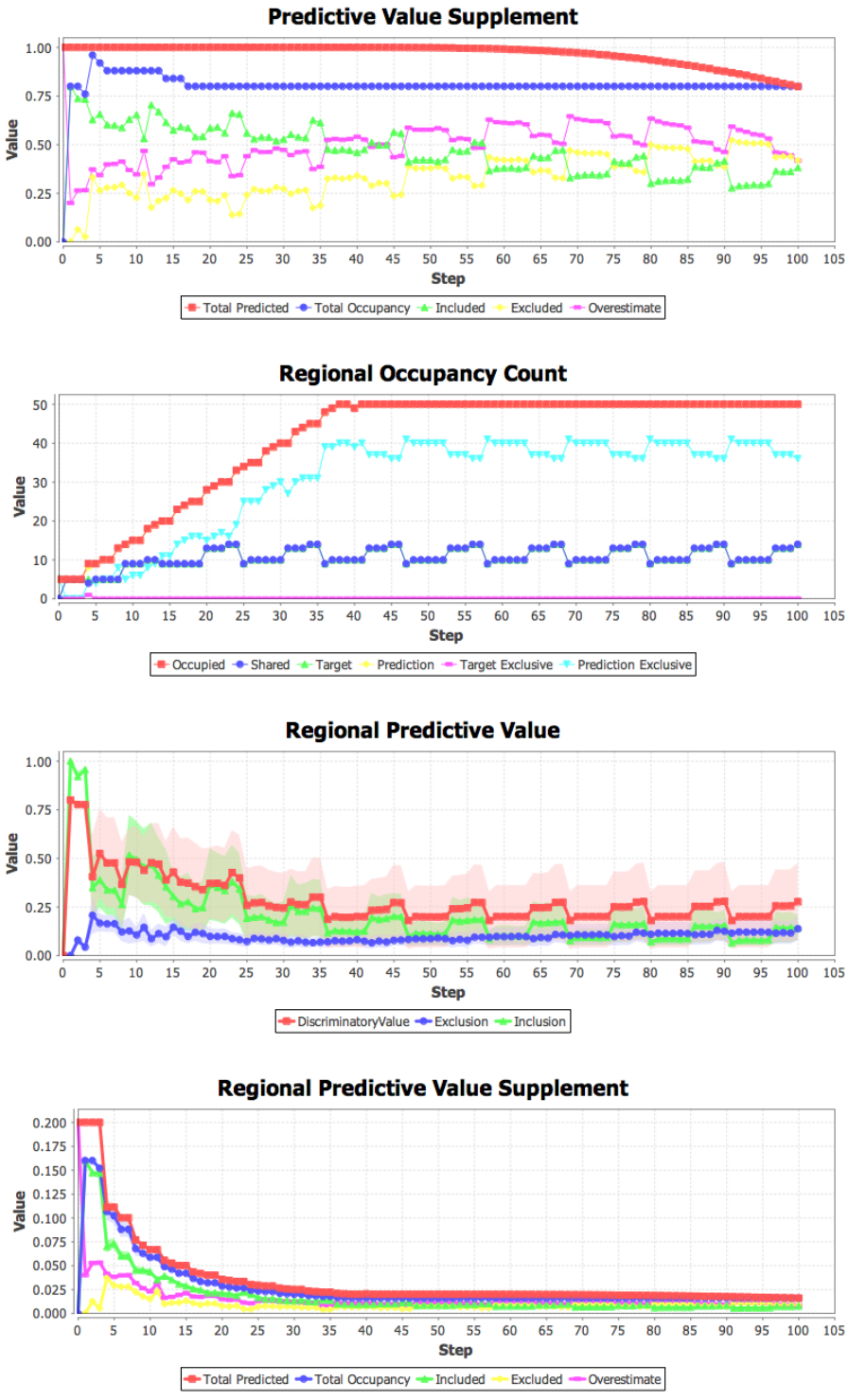

Figure C.19: *Additional predictive value plots for graph "West3 B" and batch execution "West3".*

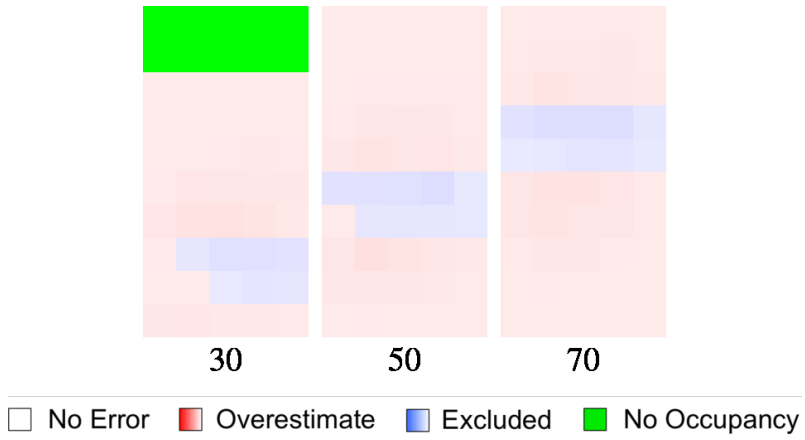

Figure C.20: *Excluded and overestimate heatmaps for graph "West3 B" and batch execution "West3".*

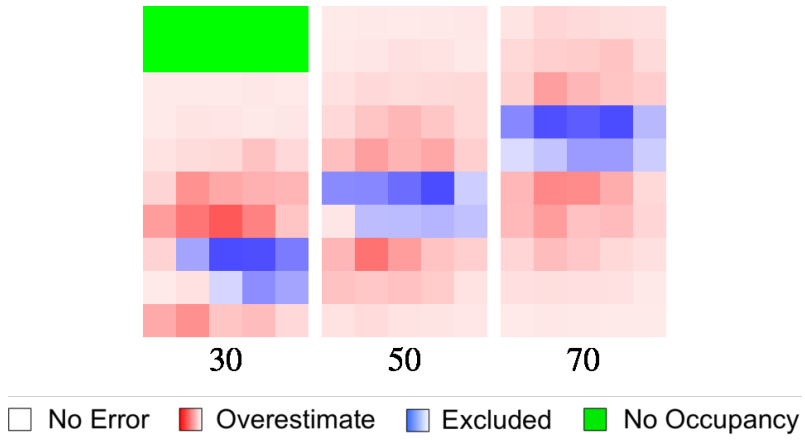

Figure C.21: *Normalized excluded and overestimate heatmaps for "West3 B" and batch execution "West3".*

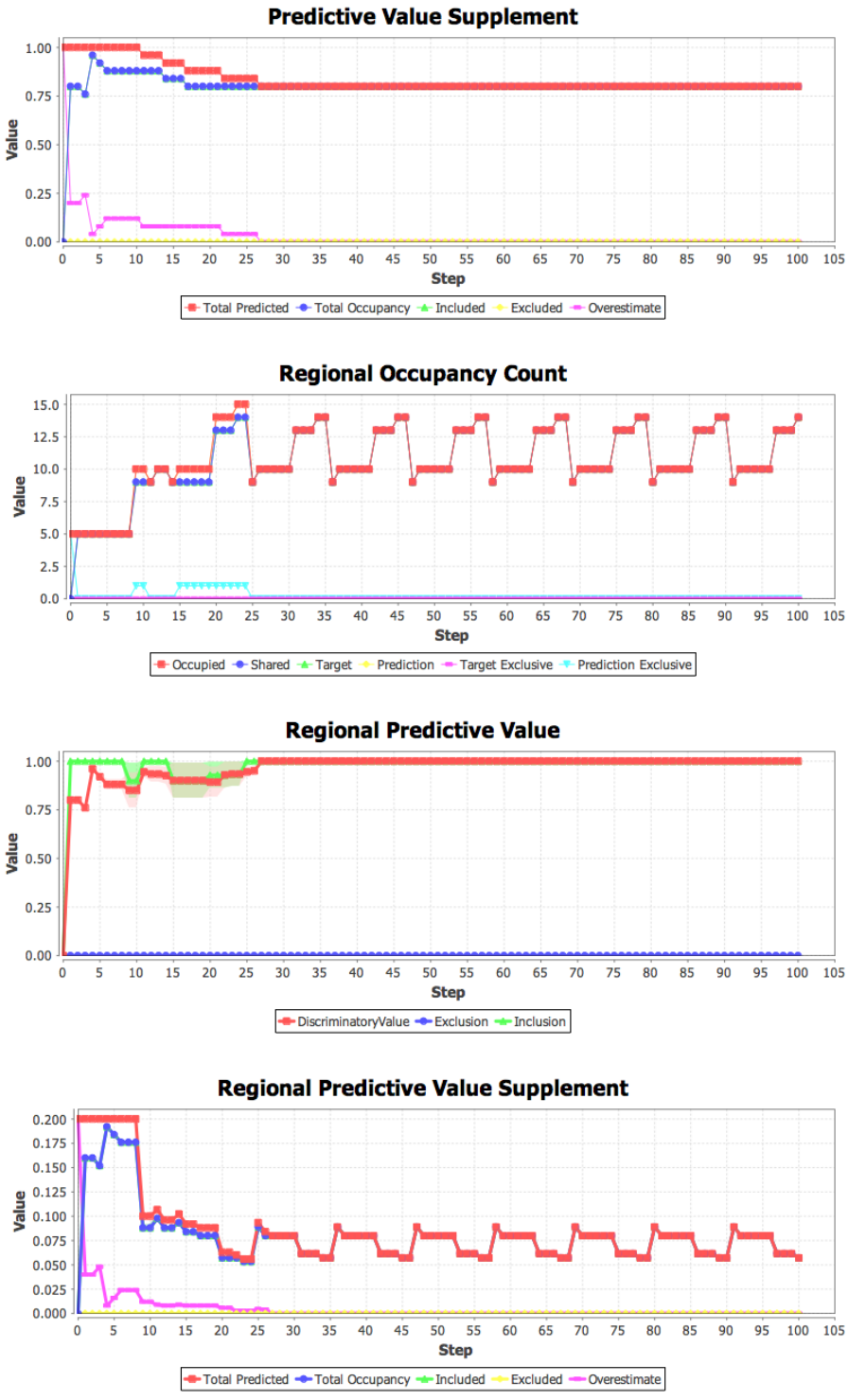

Figure C.22: *Additional predictive value plots for graph "West3 C" and batch execution "West3".*

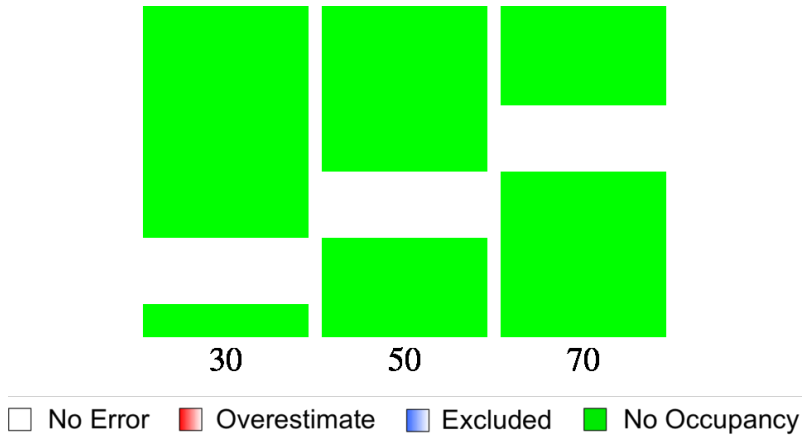

Figure C.23: *Excluded and overestimate heatmaps for graph "West3 C" and batch execution "West3".*

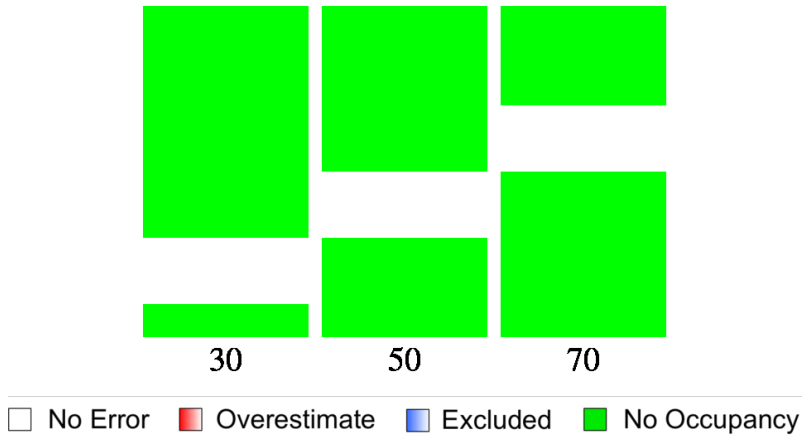

Figure C.24: *Normalized excluded and overestimate heatmaps for "West3 C" and batch execution "West3".*

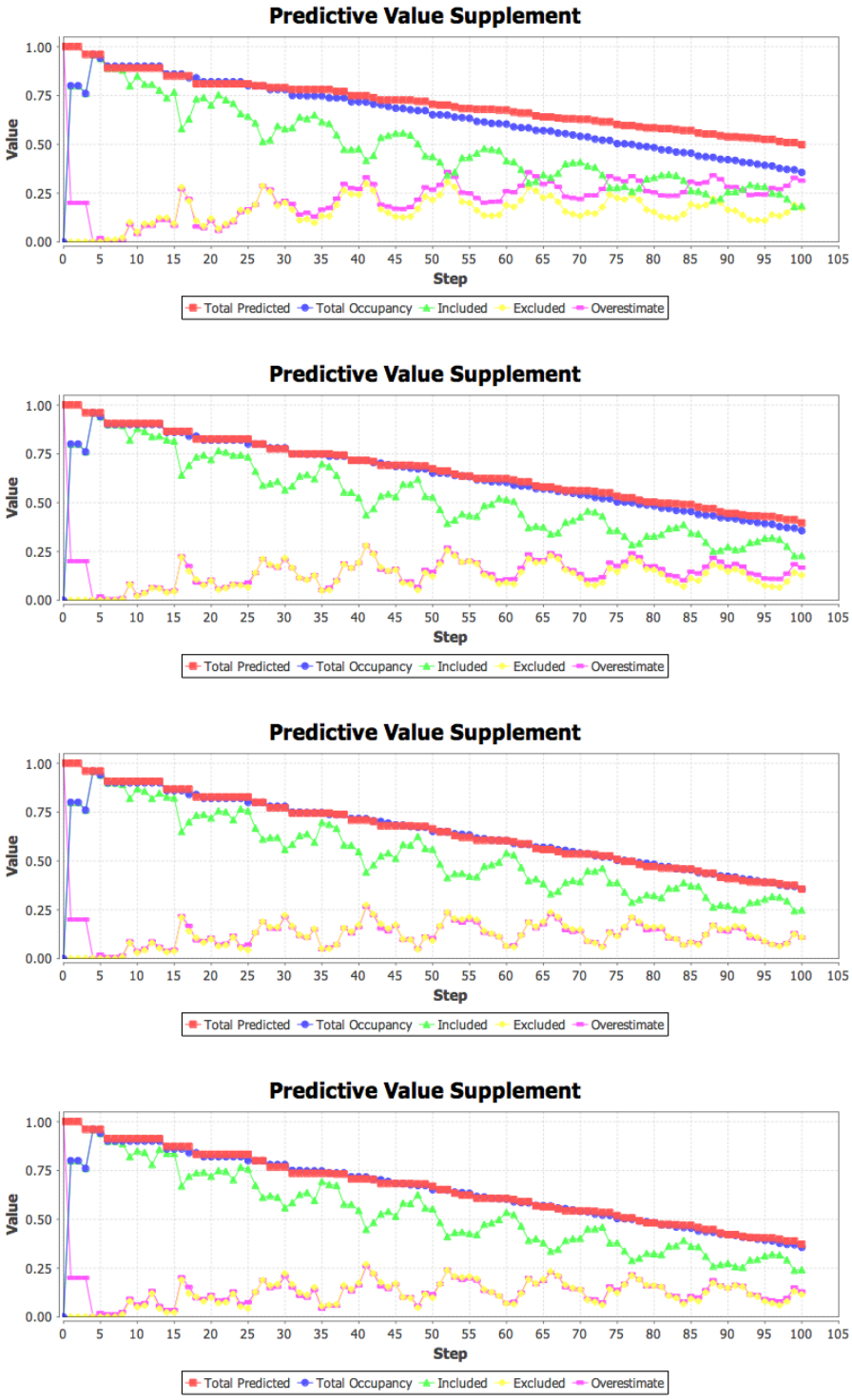

Figure C.25: *Predictive value supplement plots for graph "West4" and batch execution "West4".*

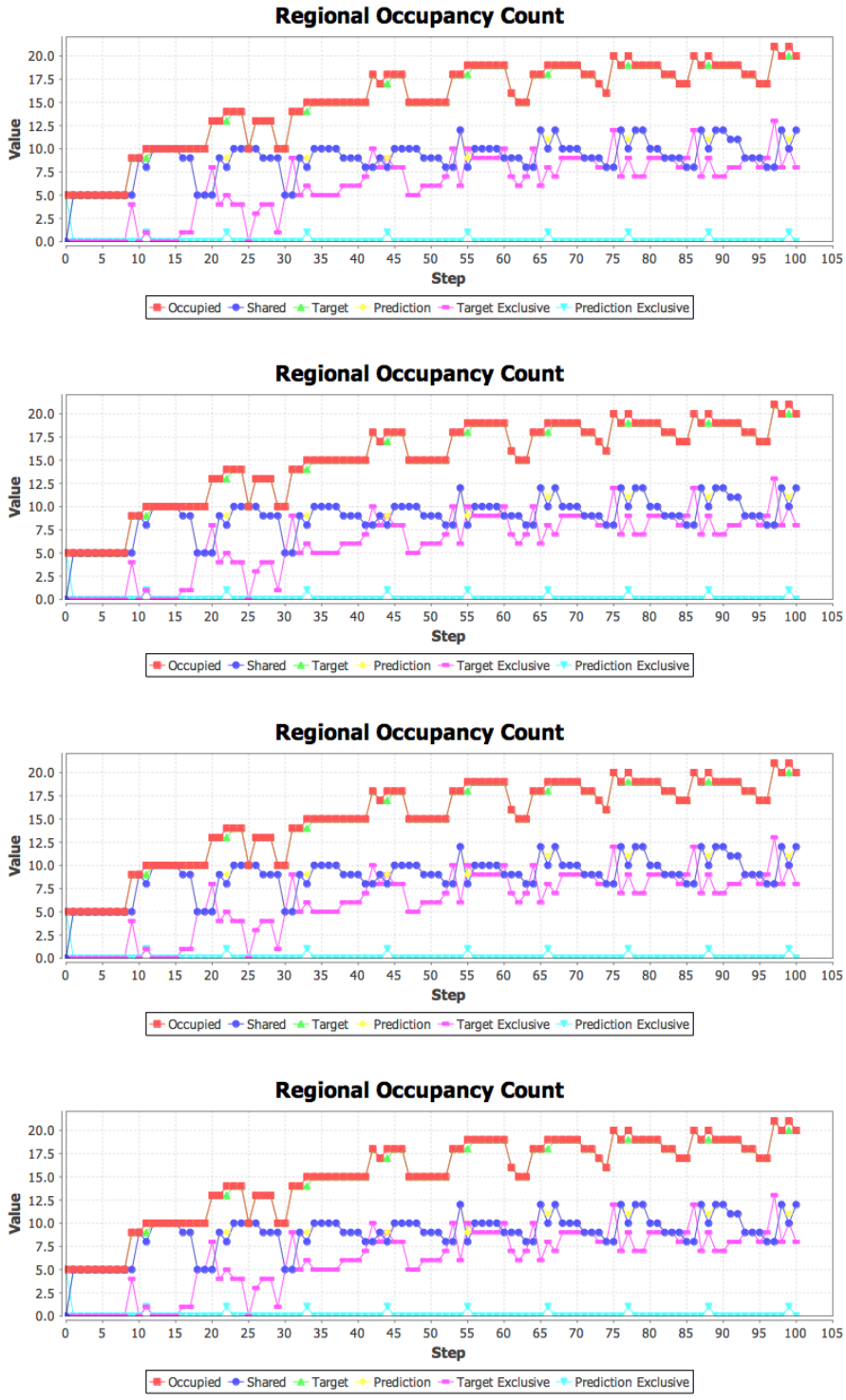

Figure C.26: *Region count plots for graph "West4" and batch execution "West4".*

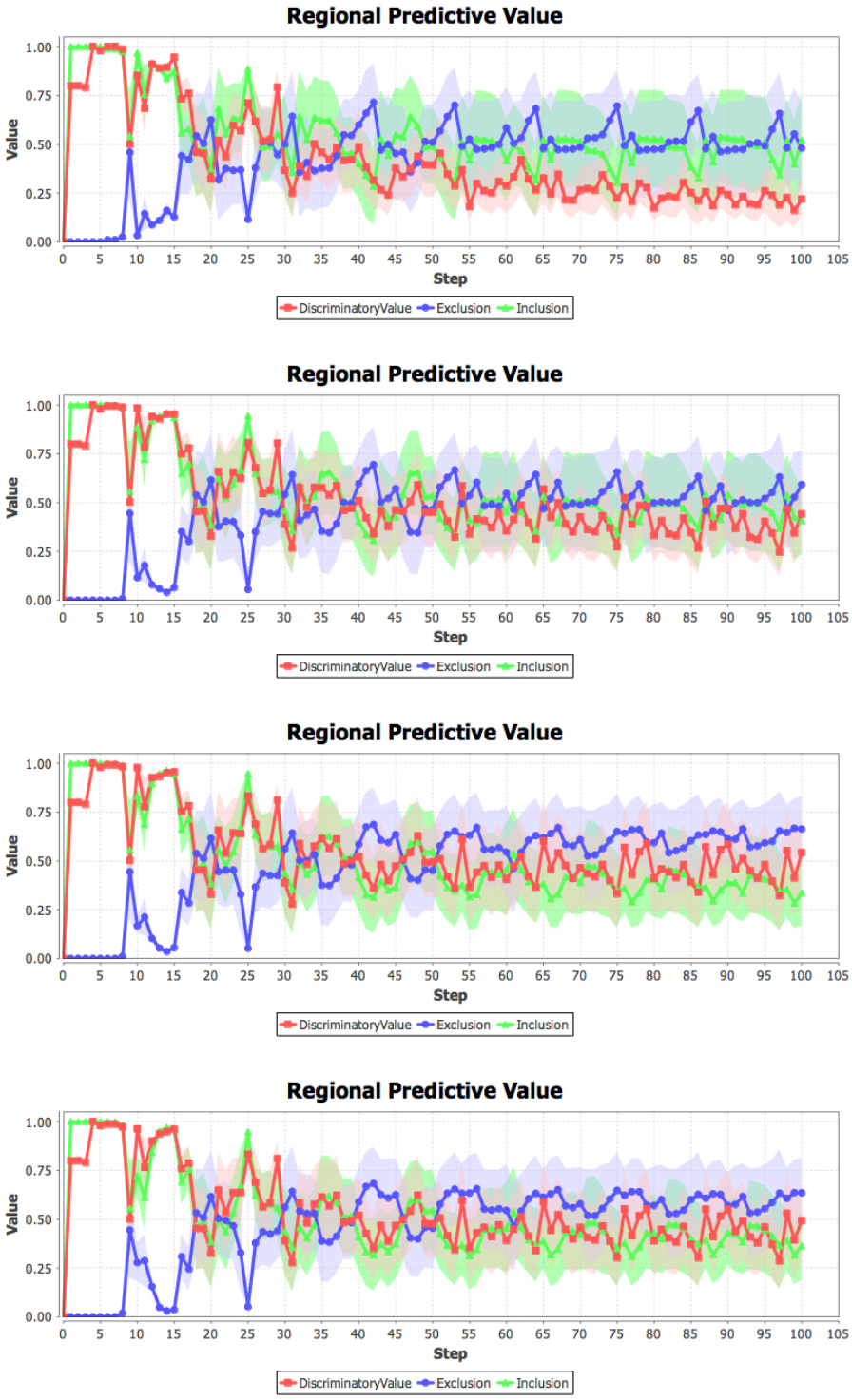

Figure C.27: *Regional predictive value plots for graph "West4" and batch execution "West4".*

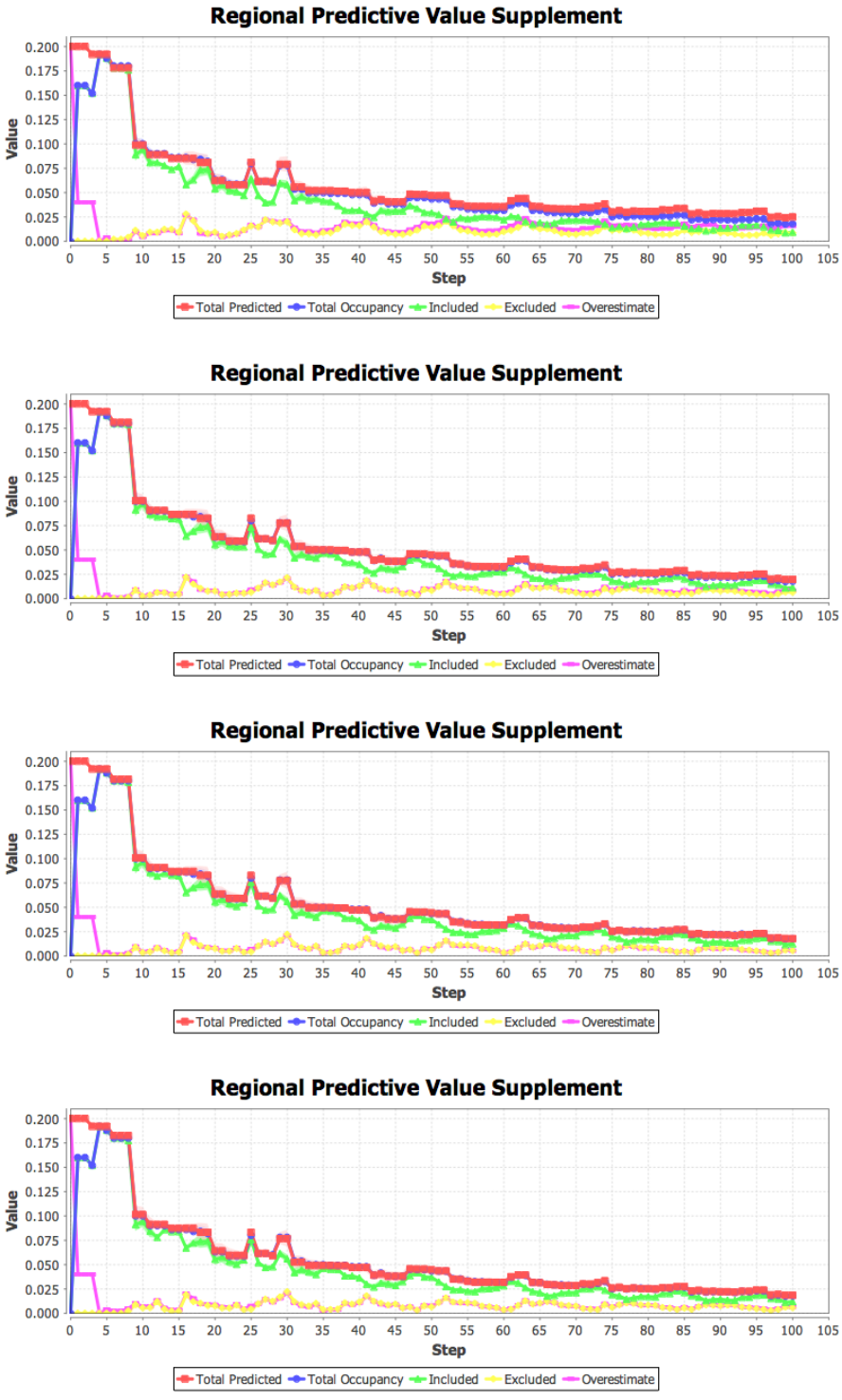

Figure C.28: *Regional predictive value supplement plots for graph "West4" and batch execution "West4".*

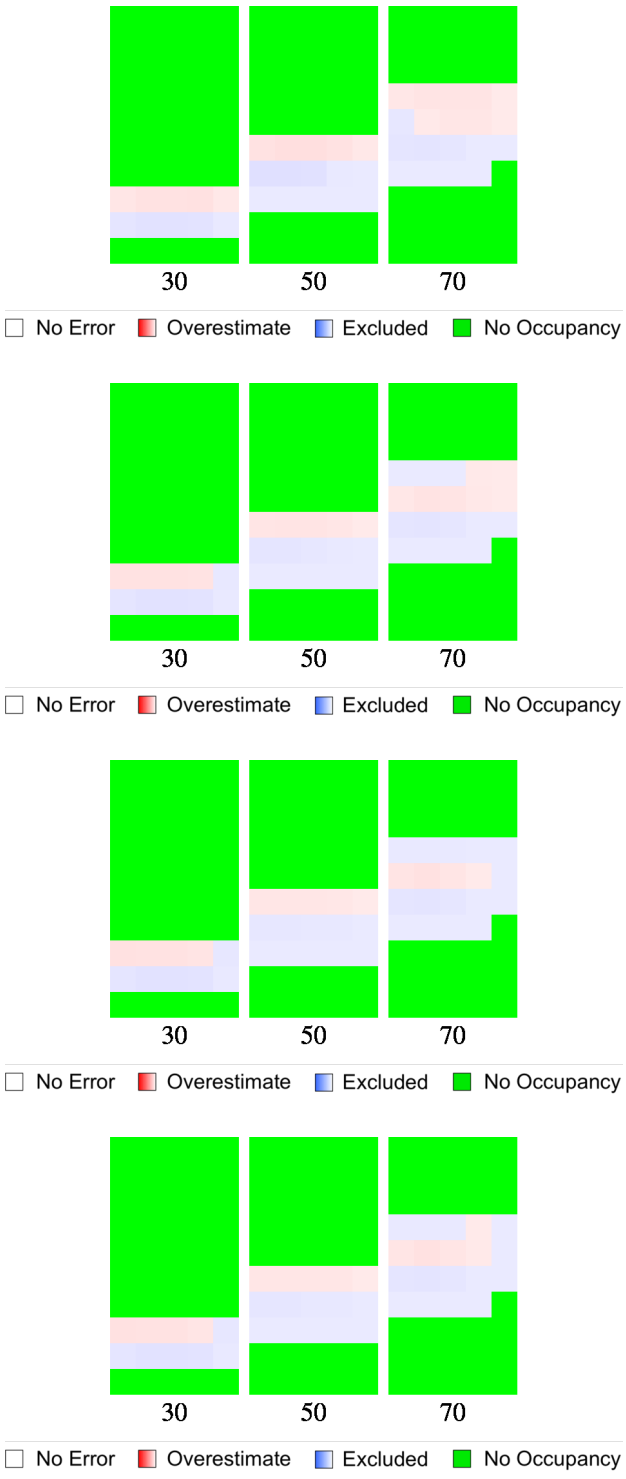

Figure C.29: *Excluded and overestimate heatmaps for graph "West4" and batch execution "West4".*

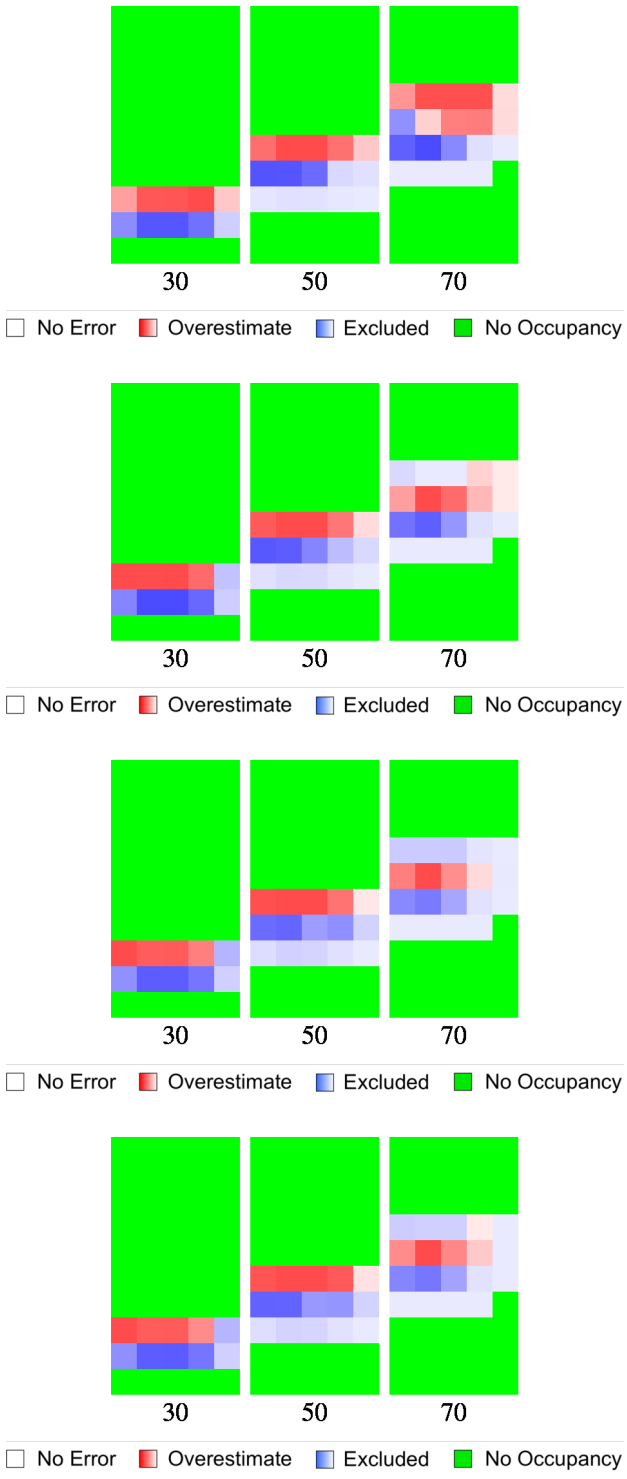

Figure C.30: *Normalized excluded and overestimate heatmaps for "West4" and batch execution "West4".*

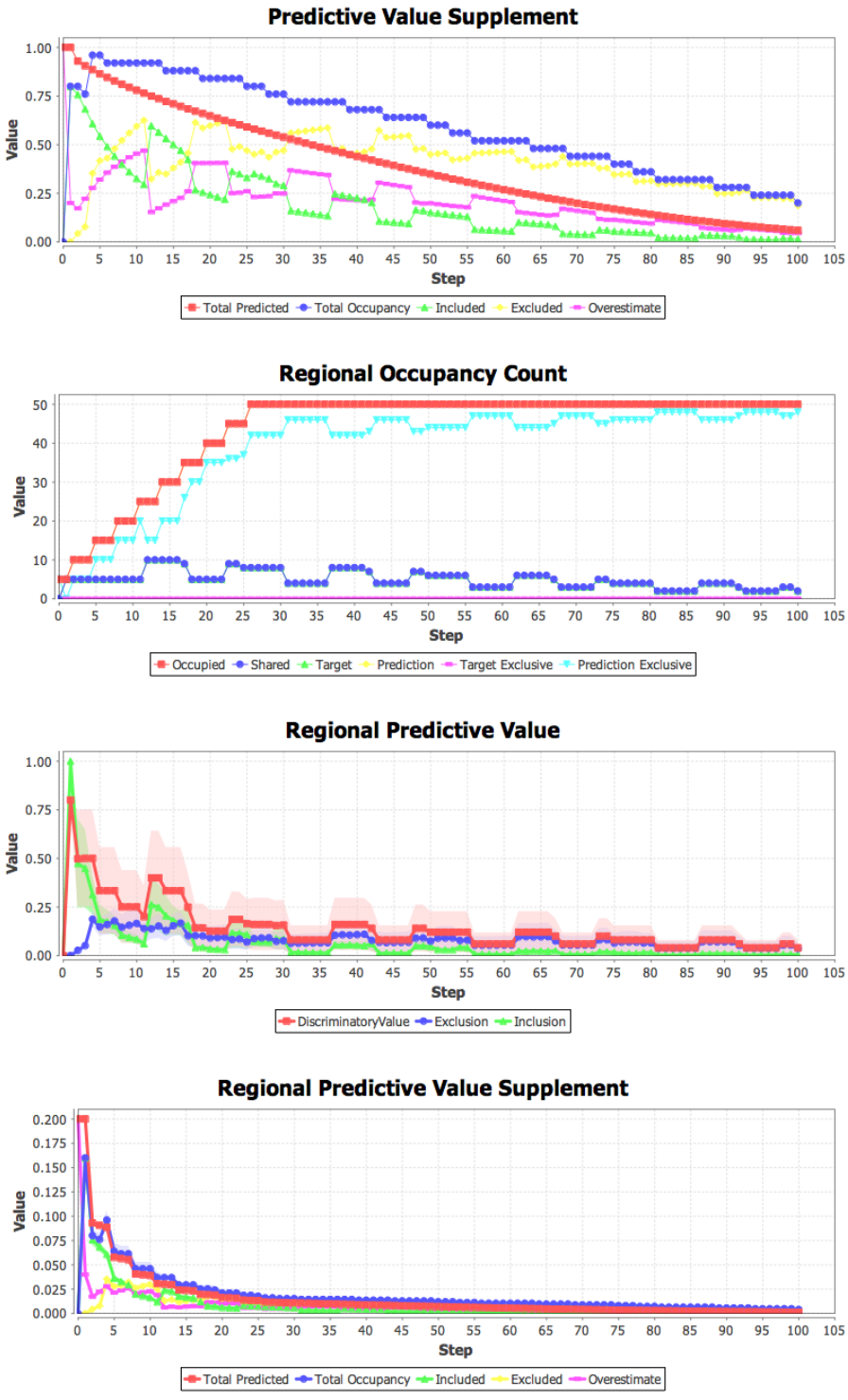

Figure C.31: *Additional predictive value plots for graph "West1 B" and batch execution "West1".*

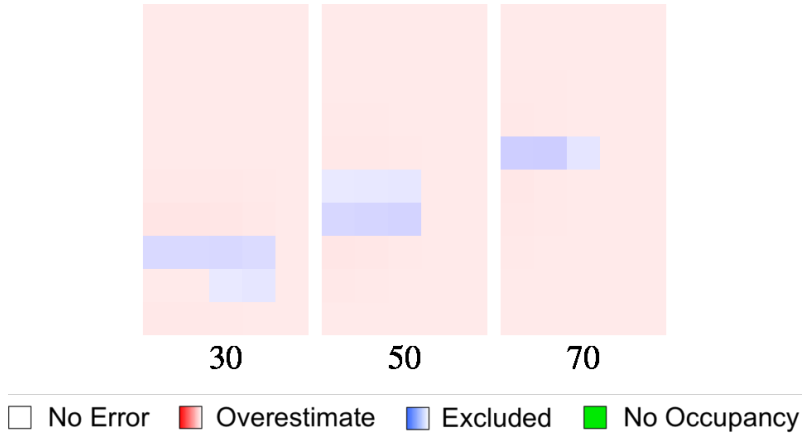

Figure C.32: *Excluded and overestimate heatmaps for graph "West1 B" and batch execution "West1".*

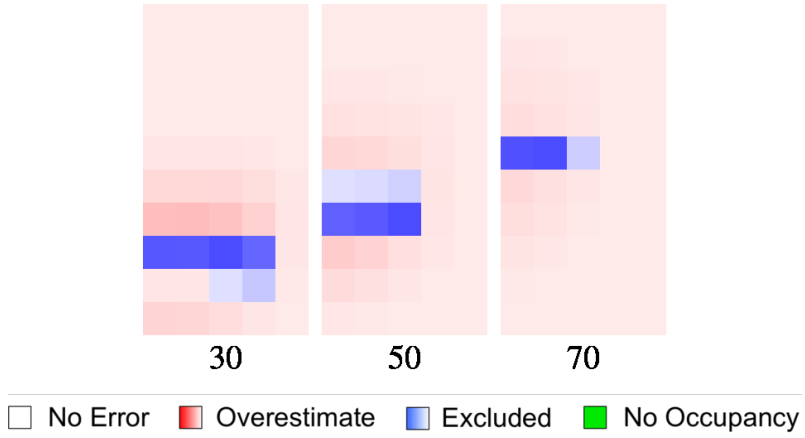

Figure C.33: *Normalized excluded and overestimate heatmaps for graph "West1 B" and batch execution "West1".*

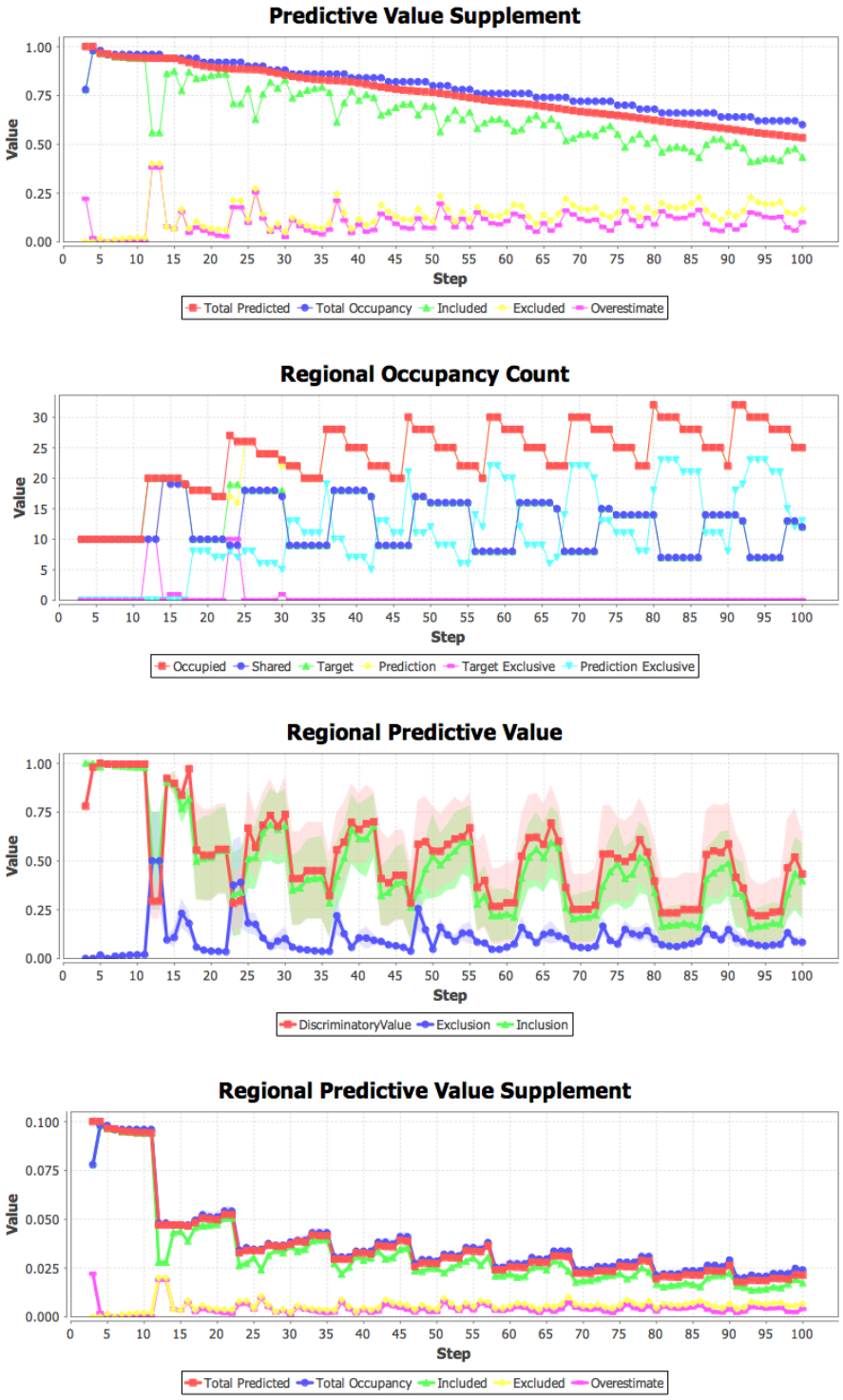

Figure C.34: *Additional predictive value plots for graph "West1 A" and batch execution "West1 Large".*

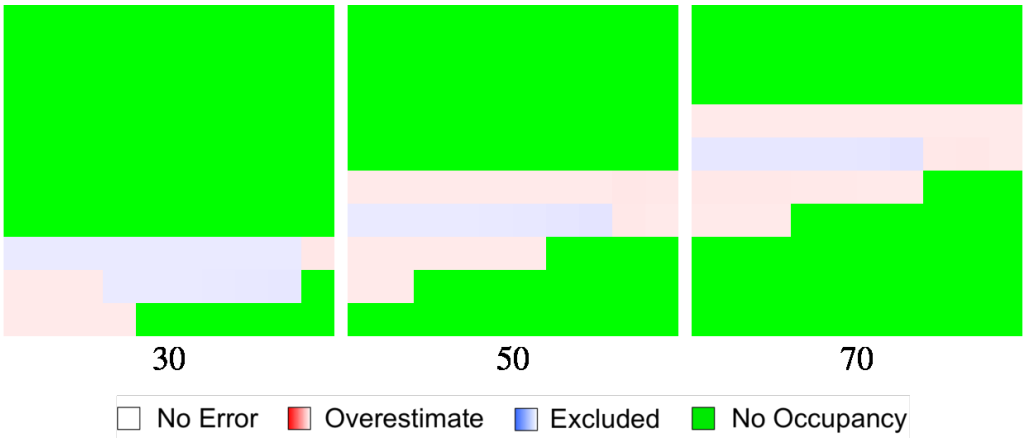

Figure C.35: *Excluded and overestimate heatmaps for graph "West1 A" and batch execution "West1 Large".*

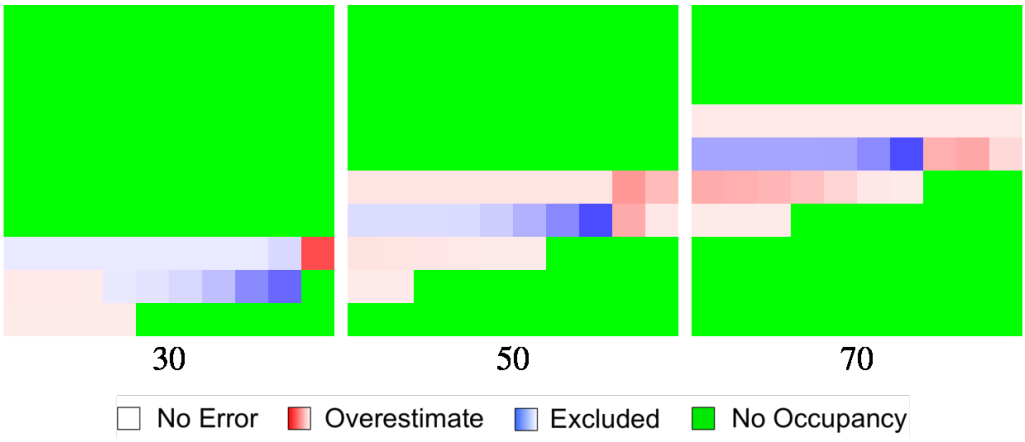

Figure C.36: *Normalized excluded and overestimate heatmaps for graph "West1 A" and batch execution "West1 Large".*

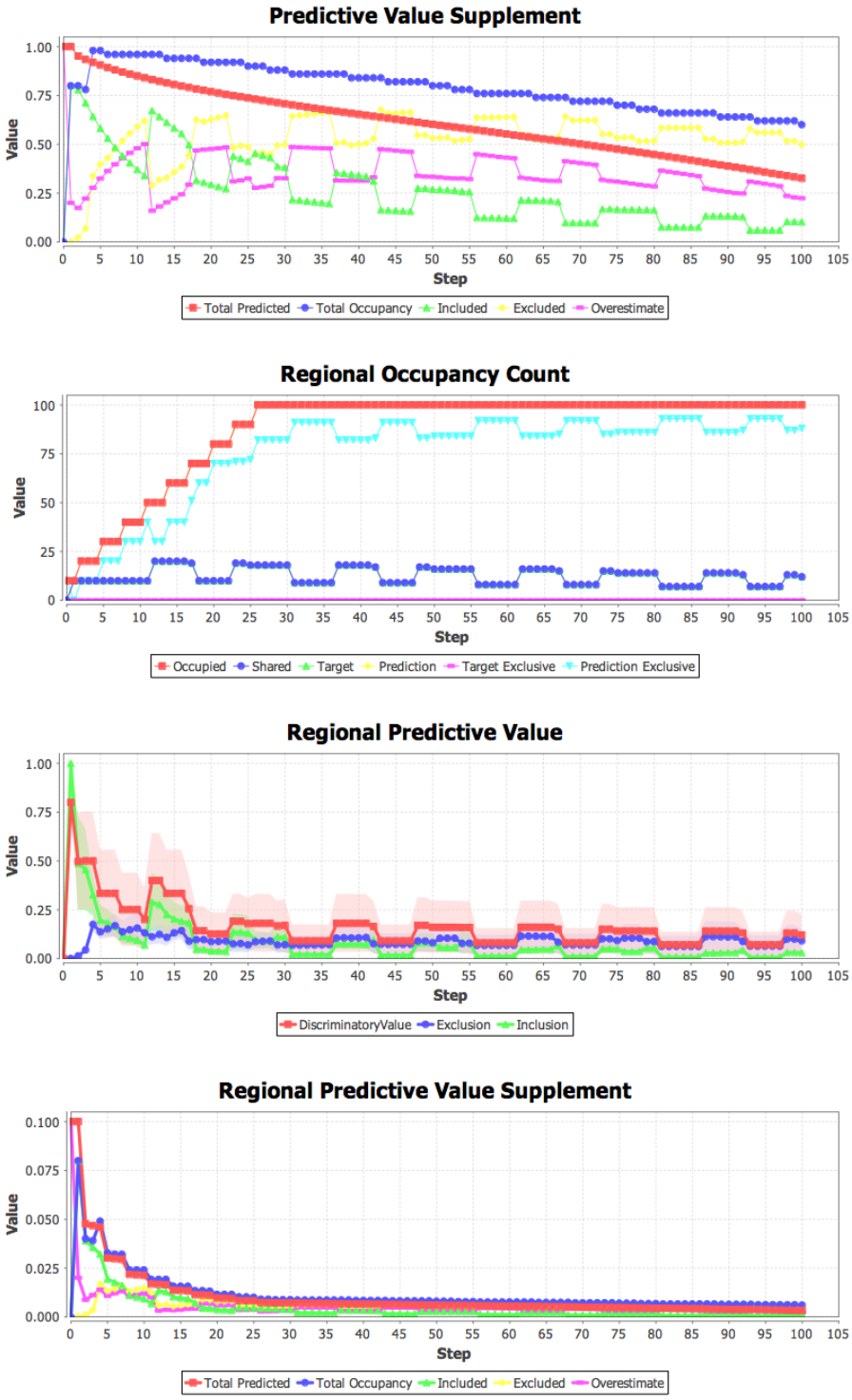

Figure C.37: *Additional predictive value plots for graph "West1 B" and batch execution "West1 Large".*

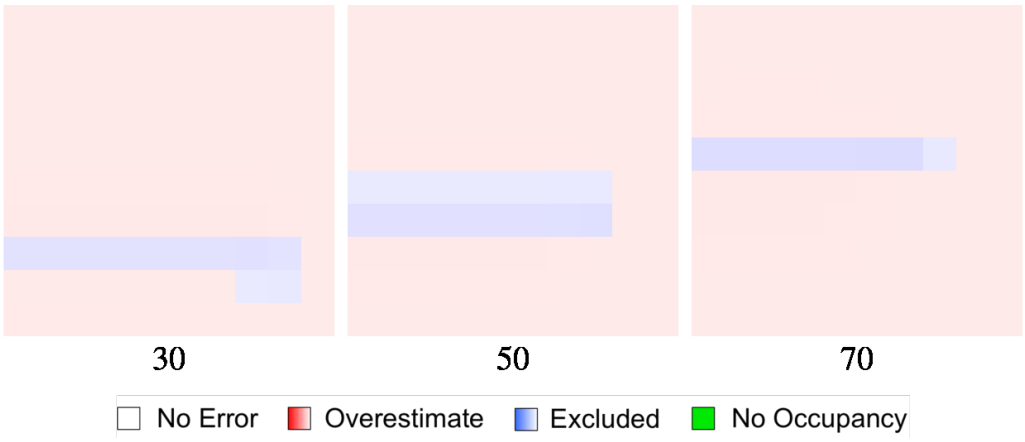

Figure C.38: *Excluded and overestimate heatmaps for graph "West1 B" and batch execution "West1 Large".*

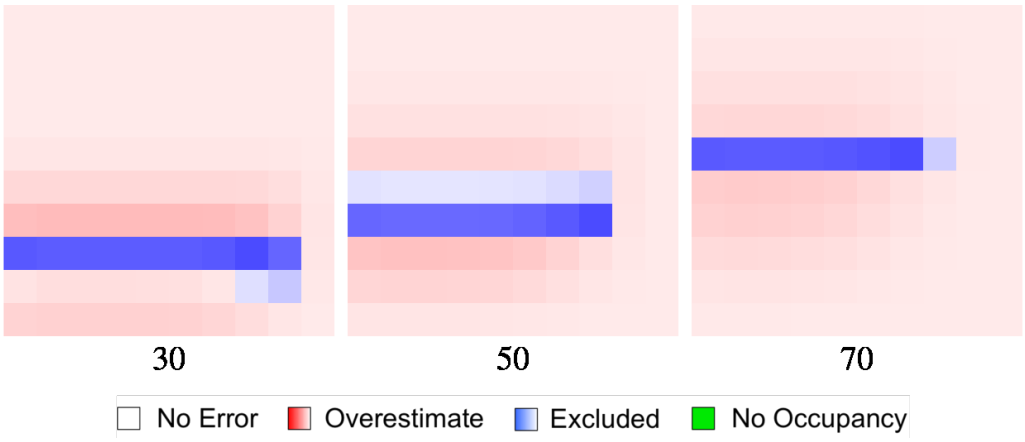

Figure C.39: *Normalized excluded and overestimate heatmaps for "West1 B" and batch execution "West1 Large".*

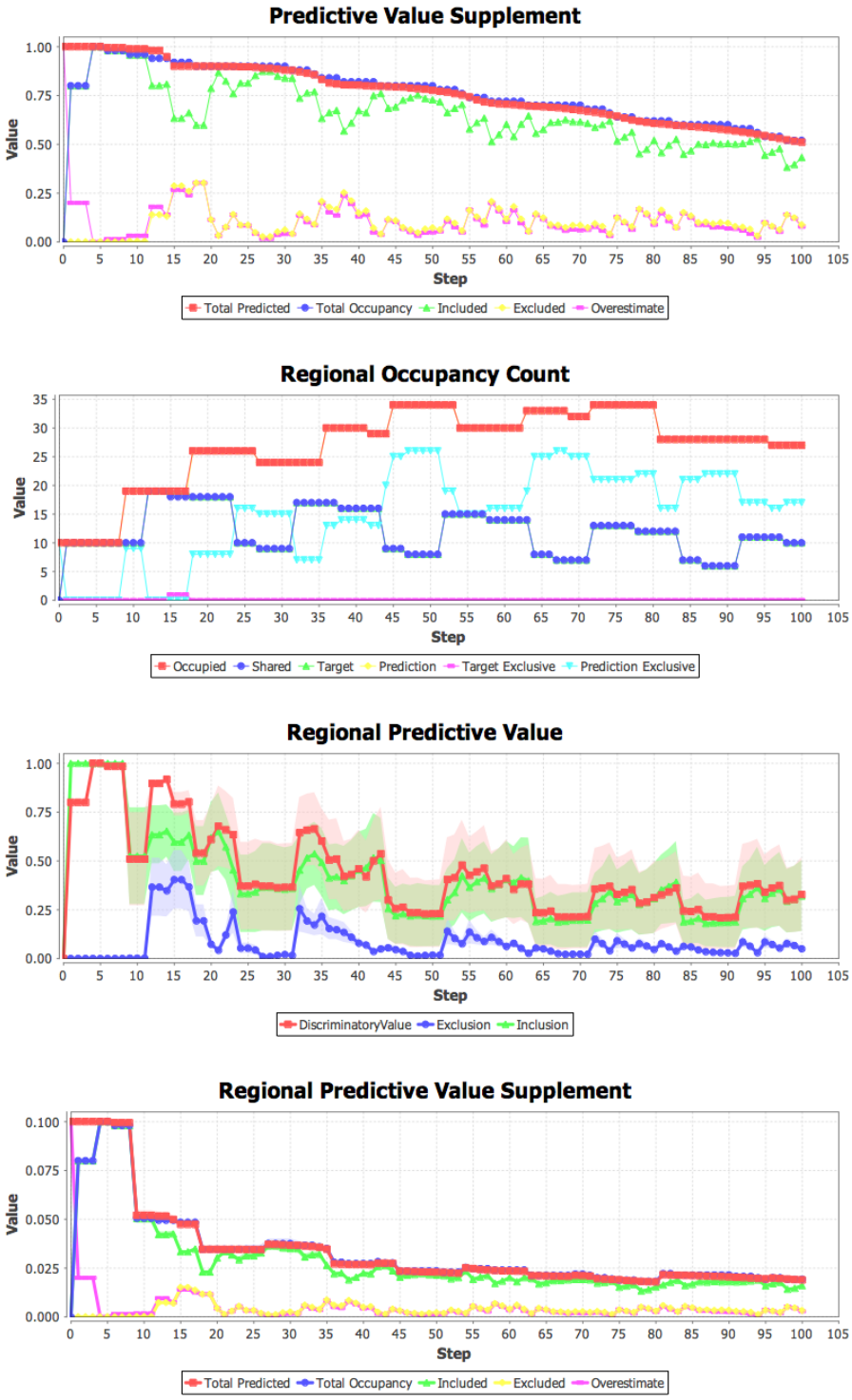

Figure C.40: *Additional predictive value plots for graph "West2 B" and batch execution "West2 Large".*

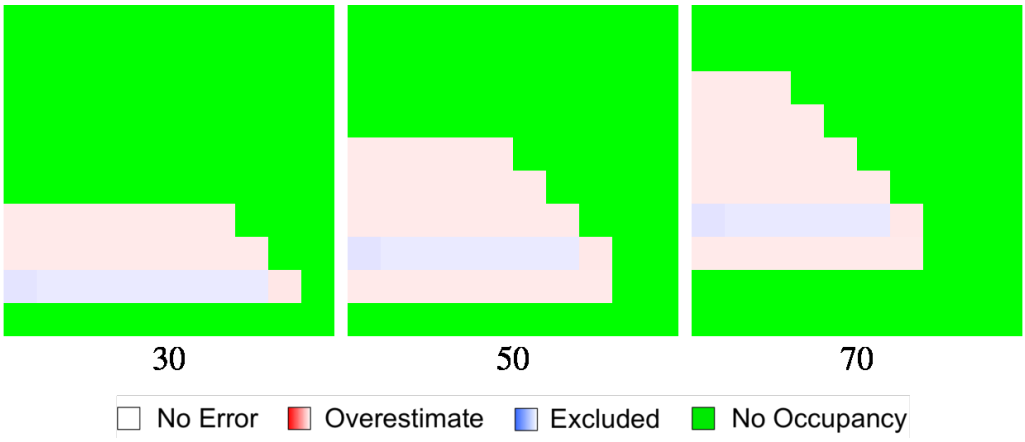

Figure C.41: *Excluded and overestimate heatmaps for graph "West2 B" and batch execution "West2 Large".*

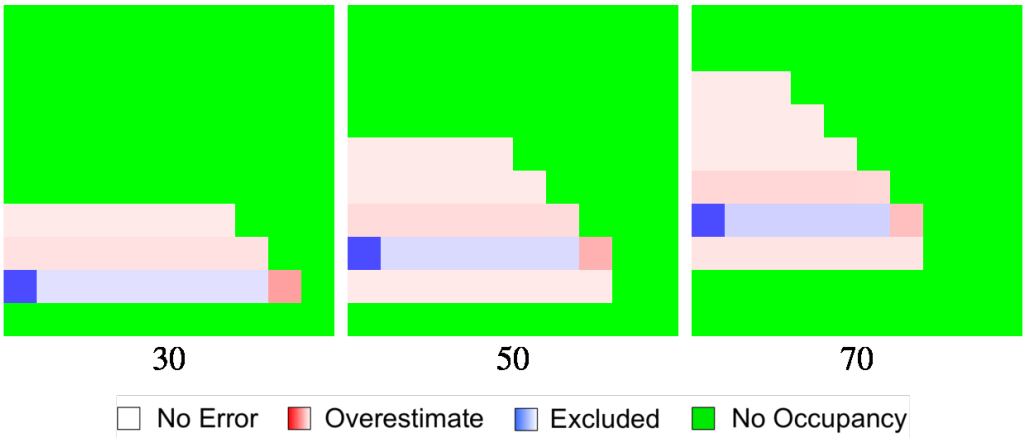

Figure C.42: *Normalized excluded and overestimate heatmaps for "West2 B" and batch execution "West2 Large".*

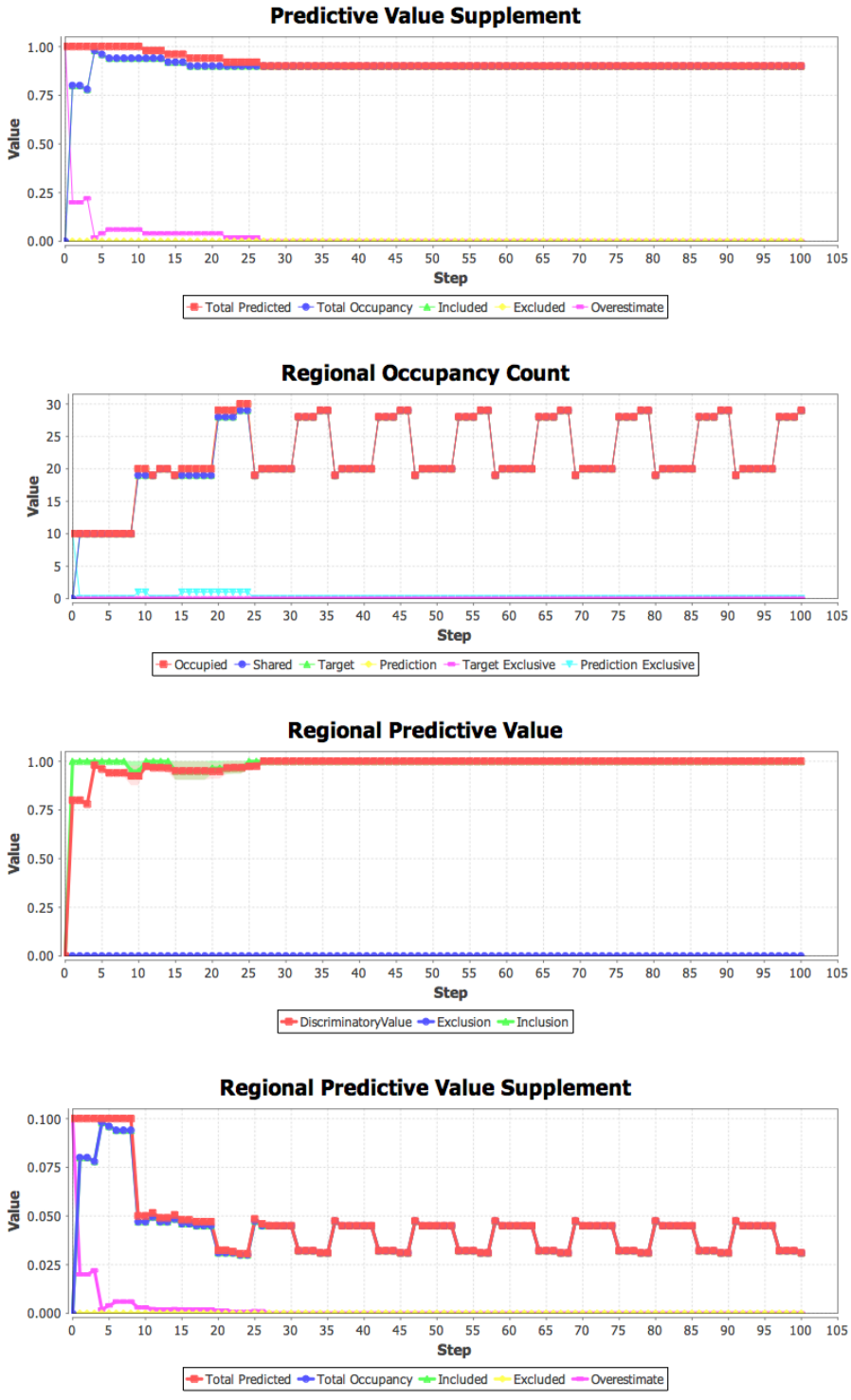

Figure C.43: *Additional predictive value plots for graph "West3 C" and batch execution "West3 Large".*

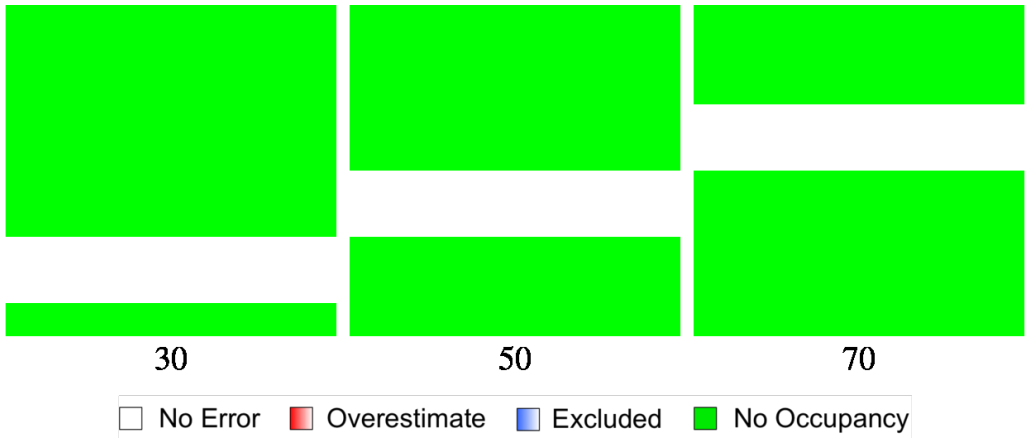

Figure C.44: *Excluded and overestimate heatmaps for graph "West3 C" and batch execution "West3 Large".*

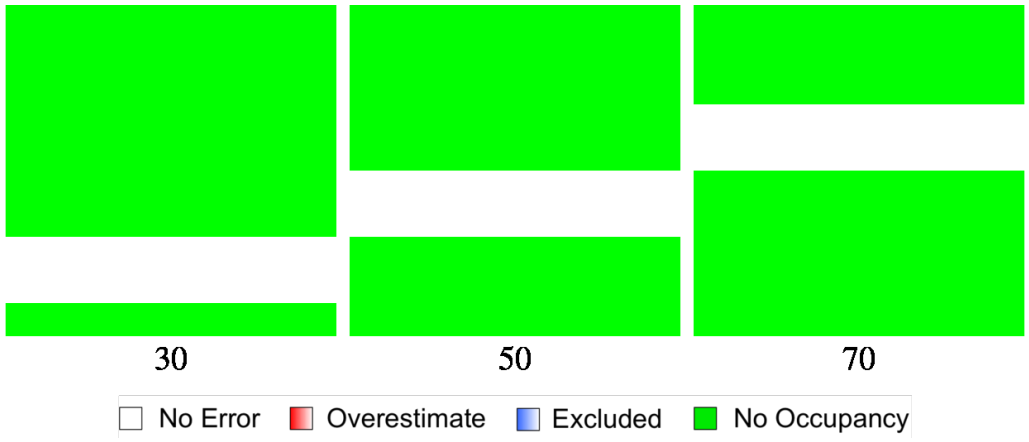

Figure C.45: *Normalized excluded and overestimate heatmaps for "West3 C" and batch execution "West3 Large".*

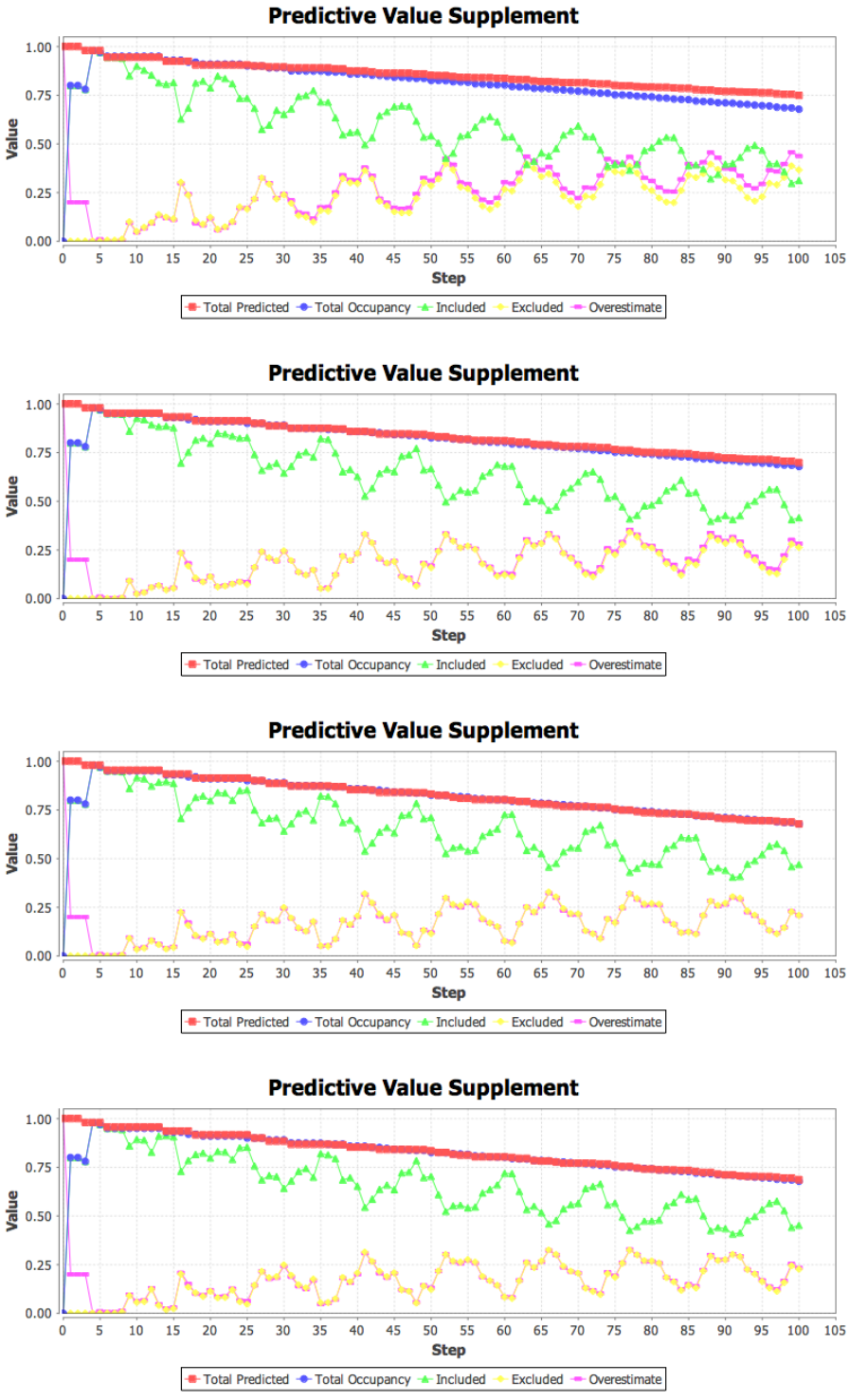

Figure C.46: *Predictive value supplement plots for graph "West4" and batch execution "West4 Large".*

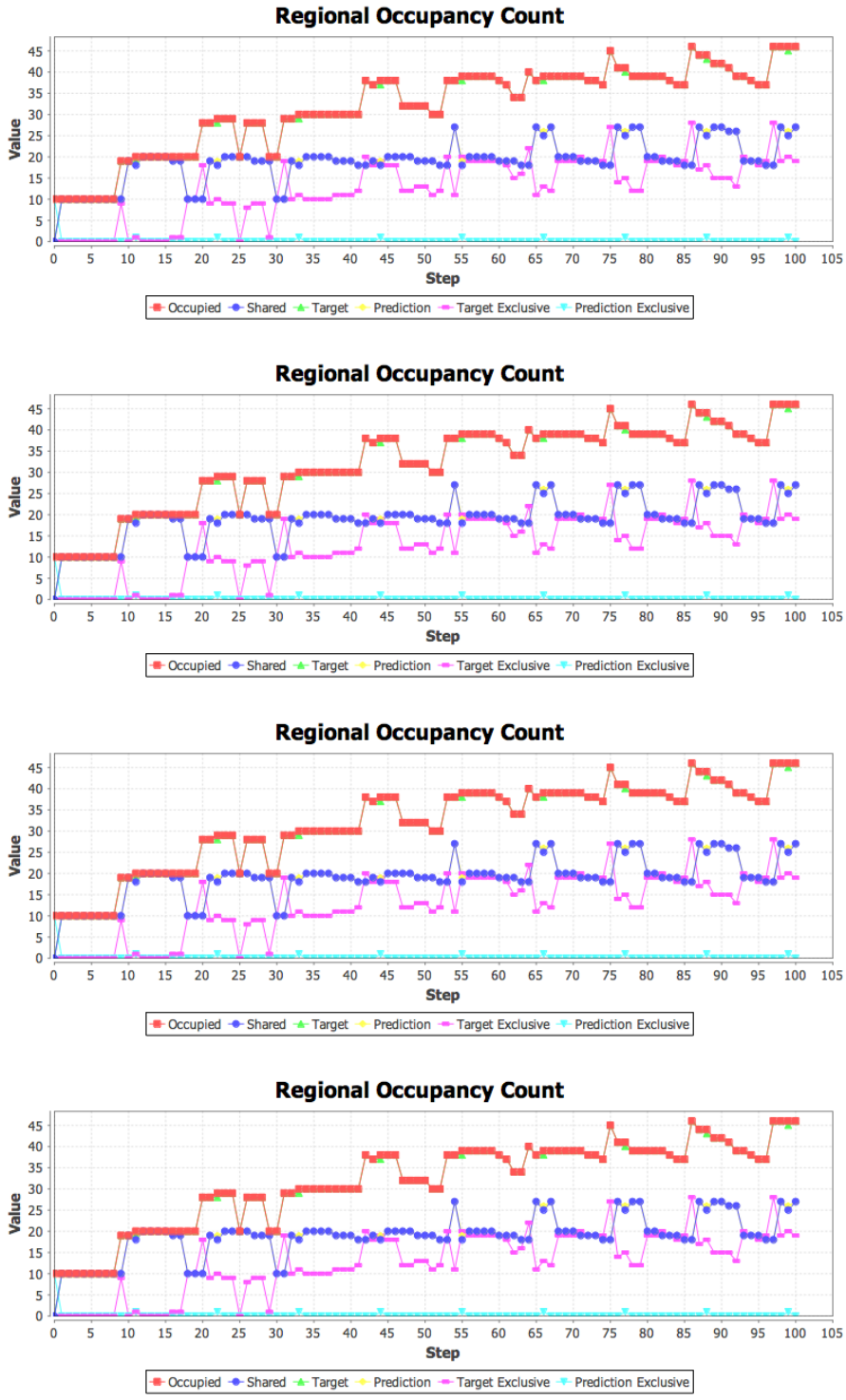

Figure C.47: *Region count plots for graph "West4" and batch execution "West4 Large".*

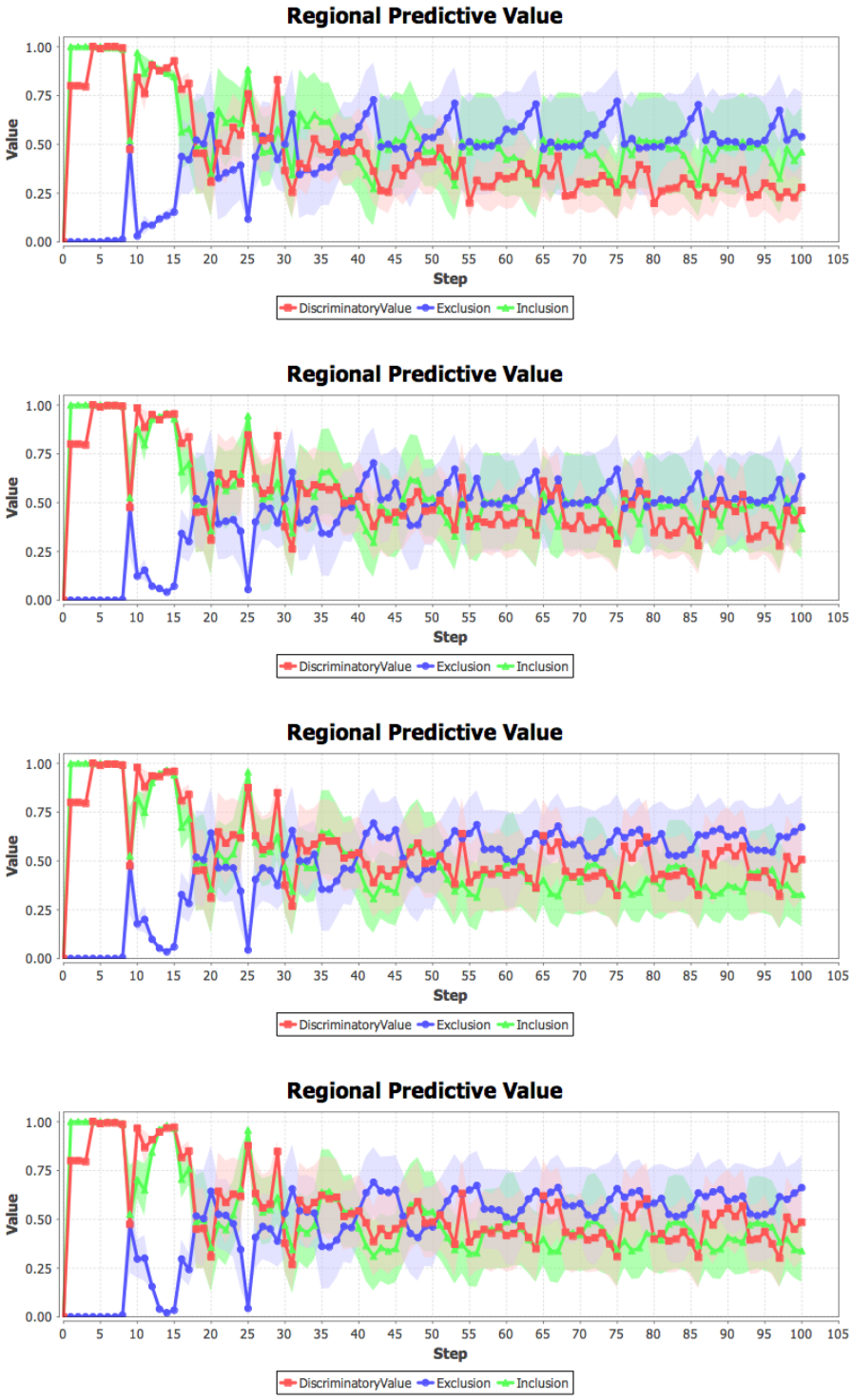

Figure C.48: *Regional predictive value plots for graph "West4" and batch execution "West4 Large".*

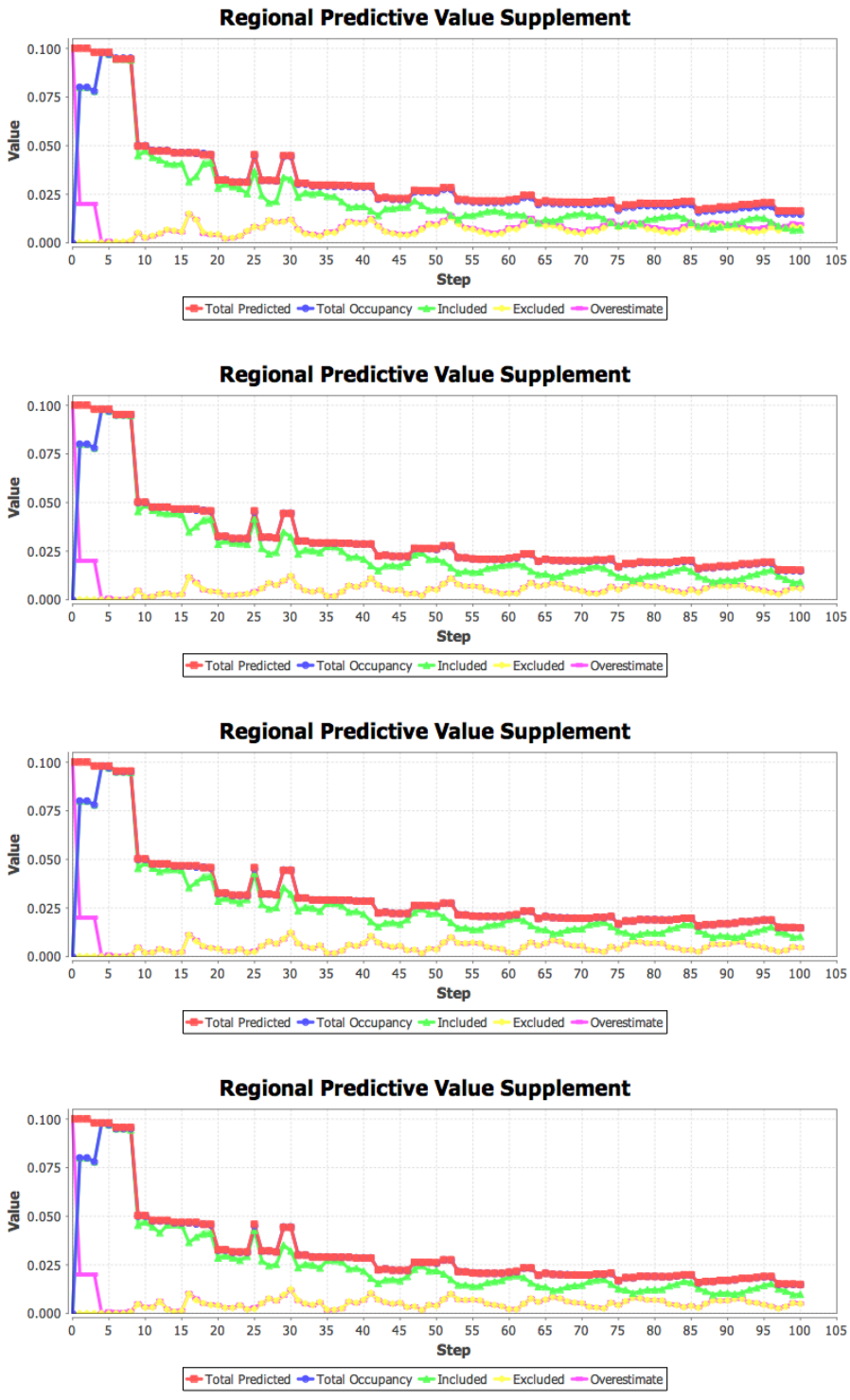

Figure C.49: *Regional predictive value supplement plots for graph "West4" and batch execution "West4 Large".*

Figure C.50: *Excluded and overestimate heatmaps for graph "West4" and batch execution "West4".*

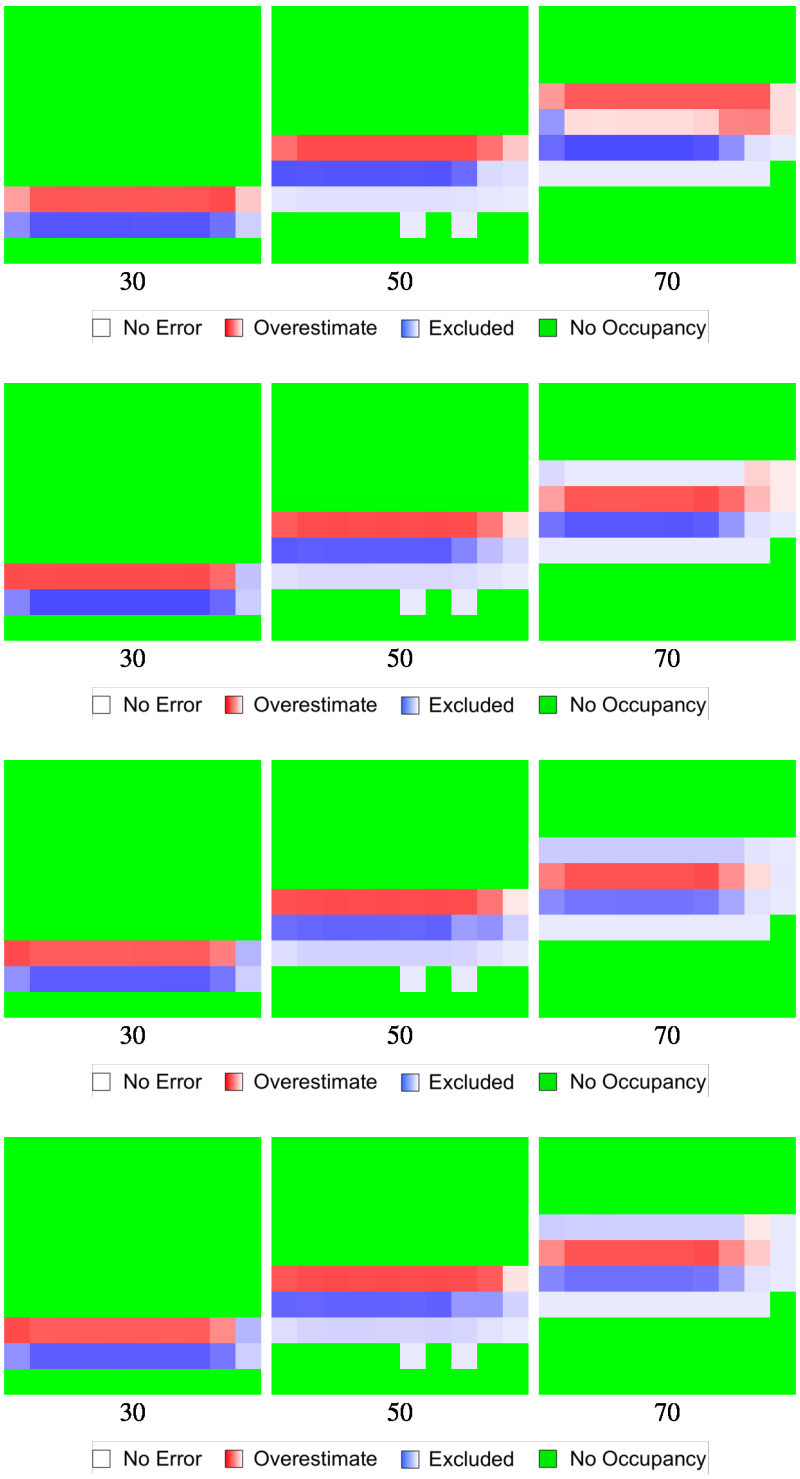

Figure C.51: *Normalized excluded and overestimate heatmaps for "West4" and batch execution "West4".*

Appendix D

Supplemental Sensitivity and Bias Visuals

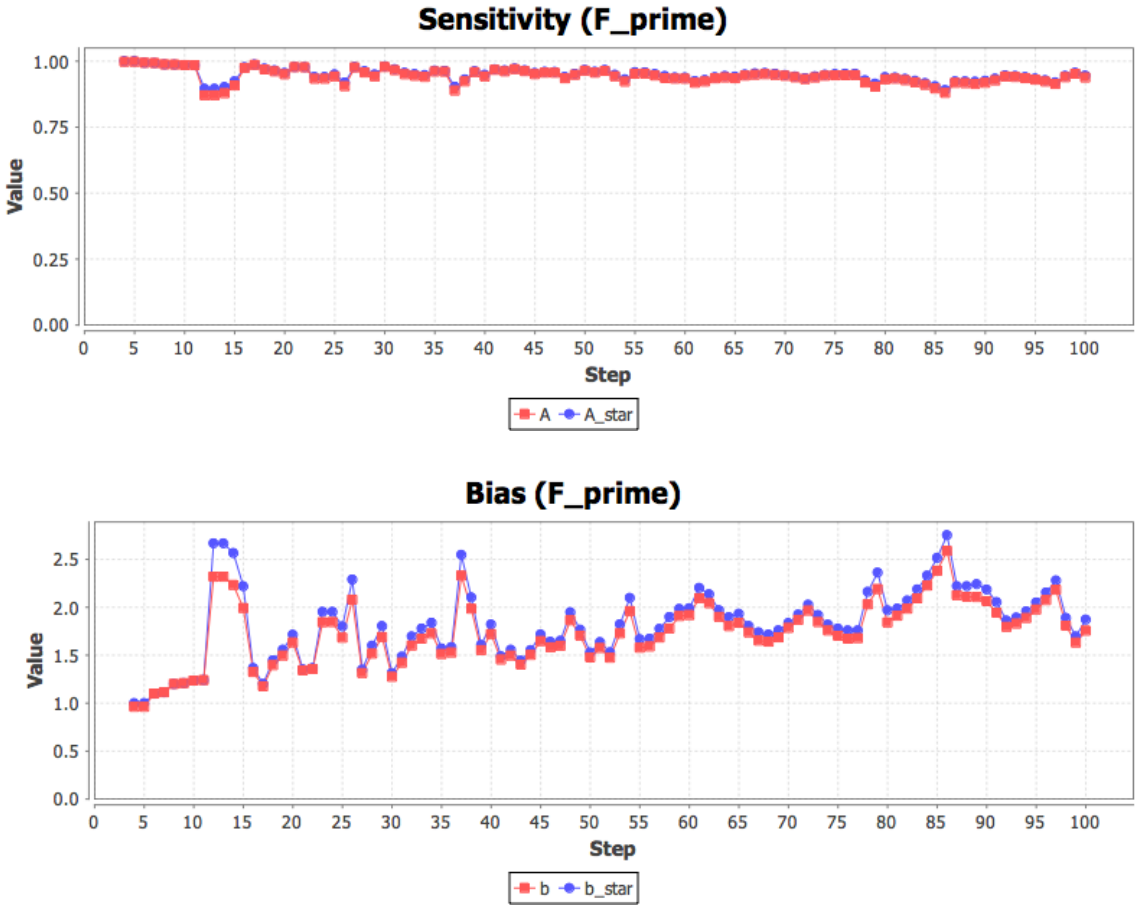

Figure D.1: *Sensitivity and bias using F' for graph "East".*

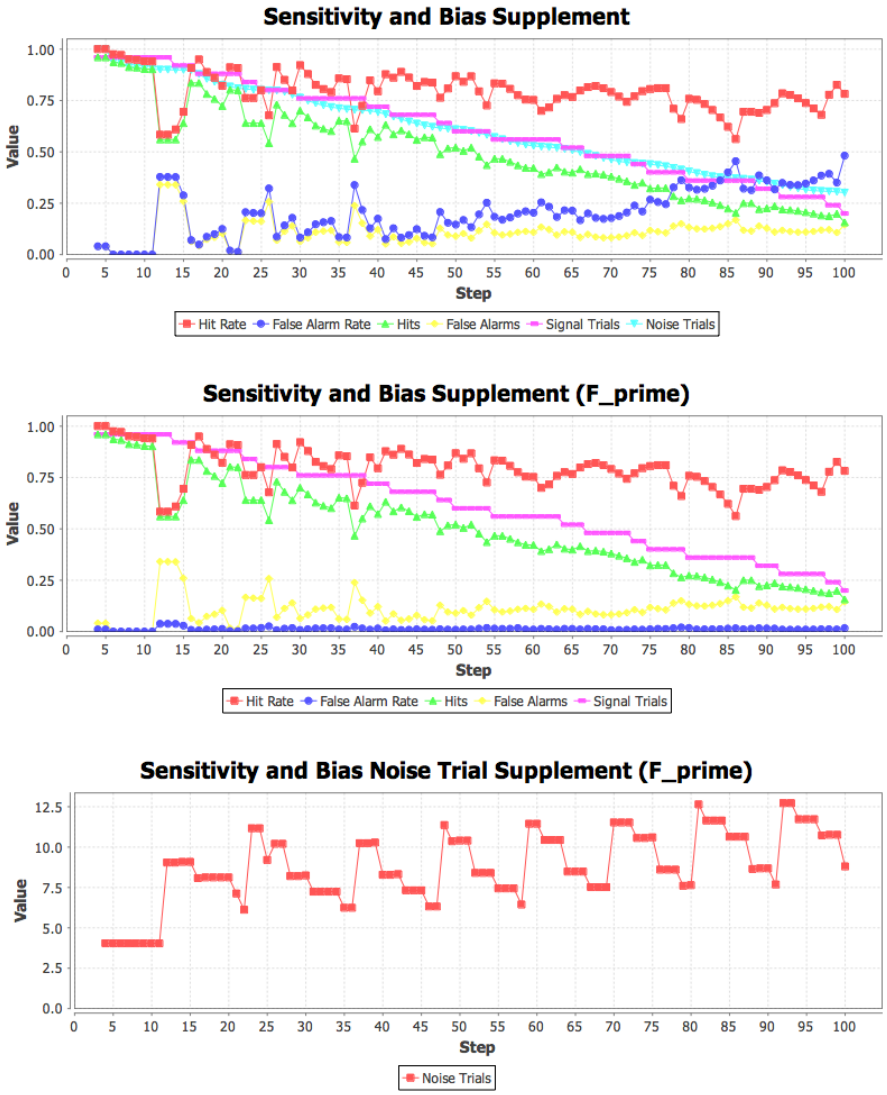

Figure D.2: *Additional sensitivity and bias plots for graph "East" and batch execution "East".*

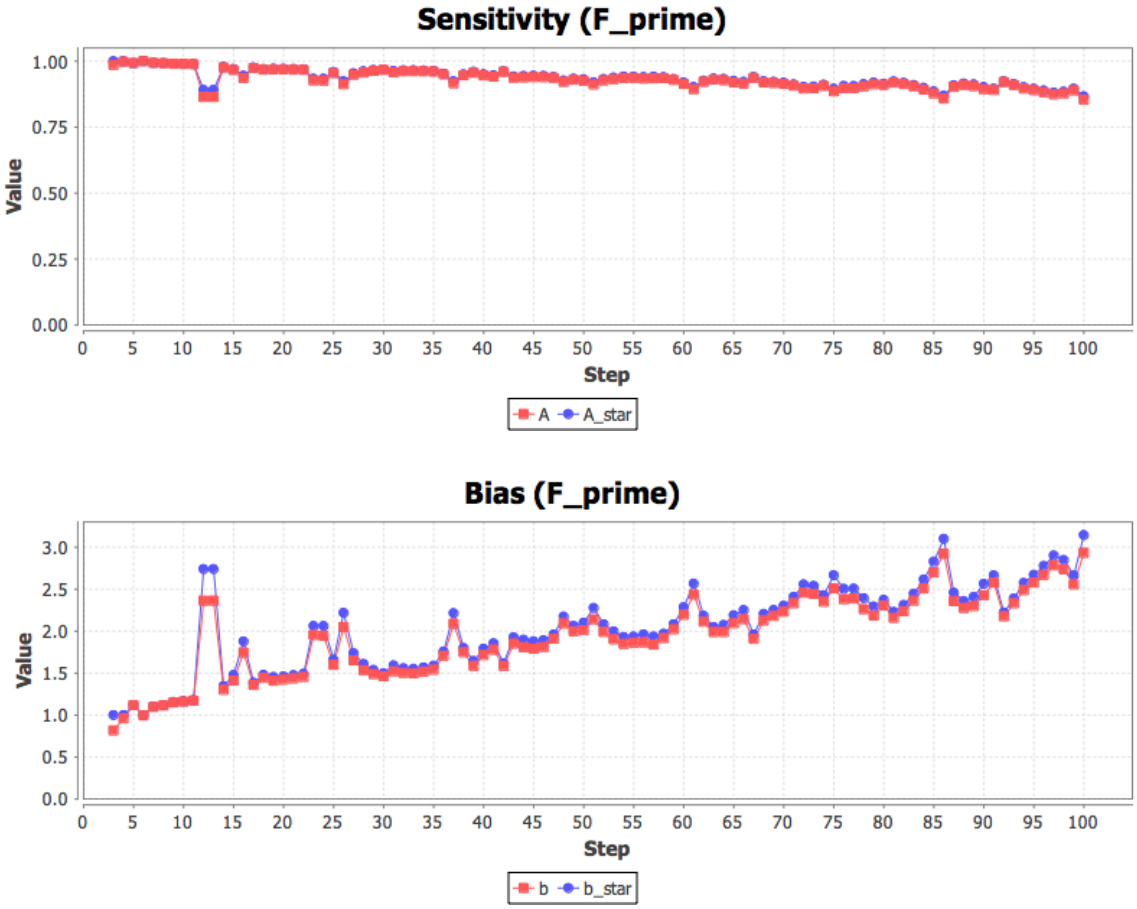

Figure D.3: *Sensitivity and bias using F' for graph "West1 A".*

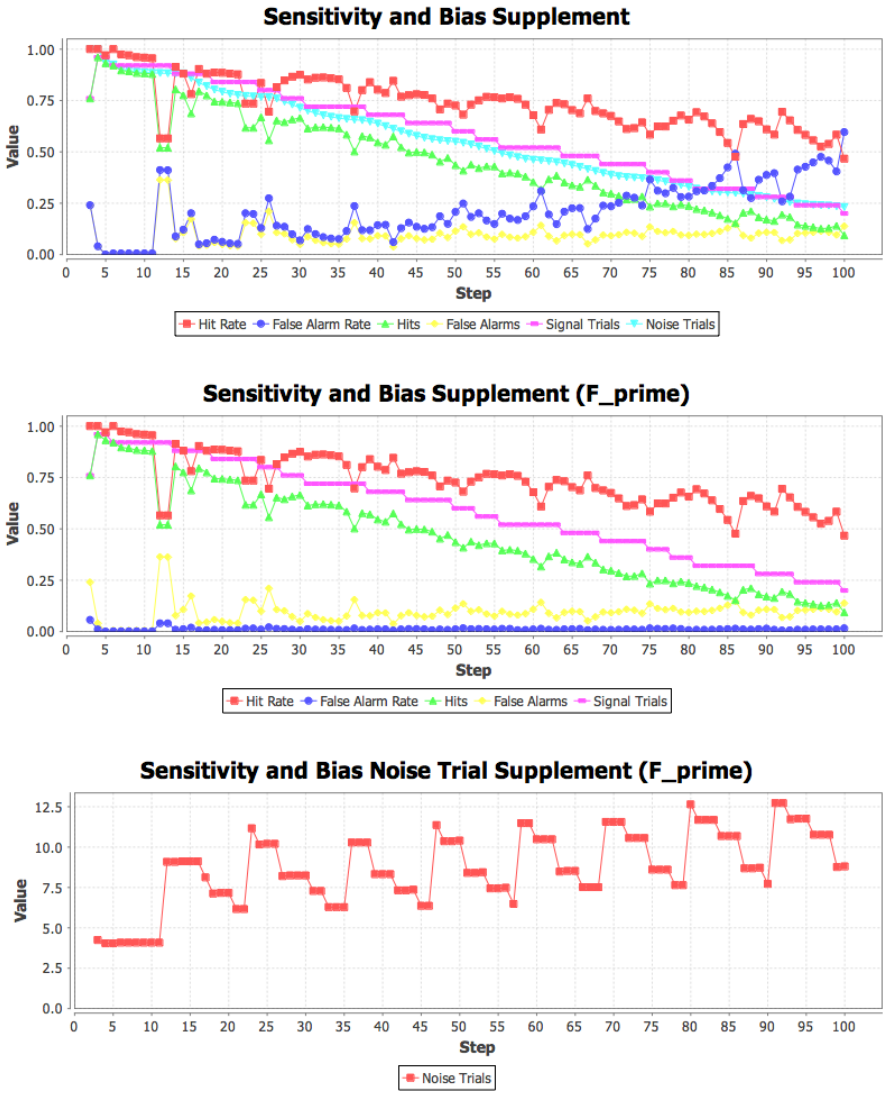

Figure D.4: *Additional sensitivity and bias plots for graph "West1 A" and batch execution "West1".*

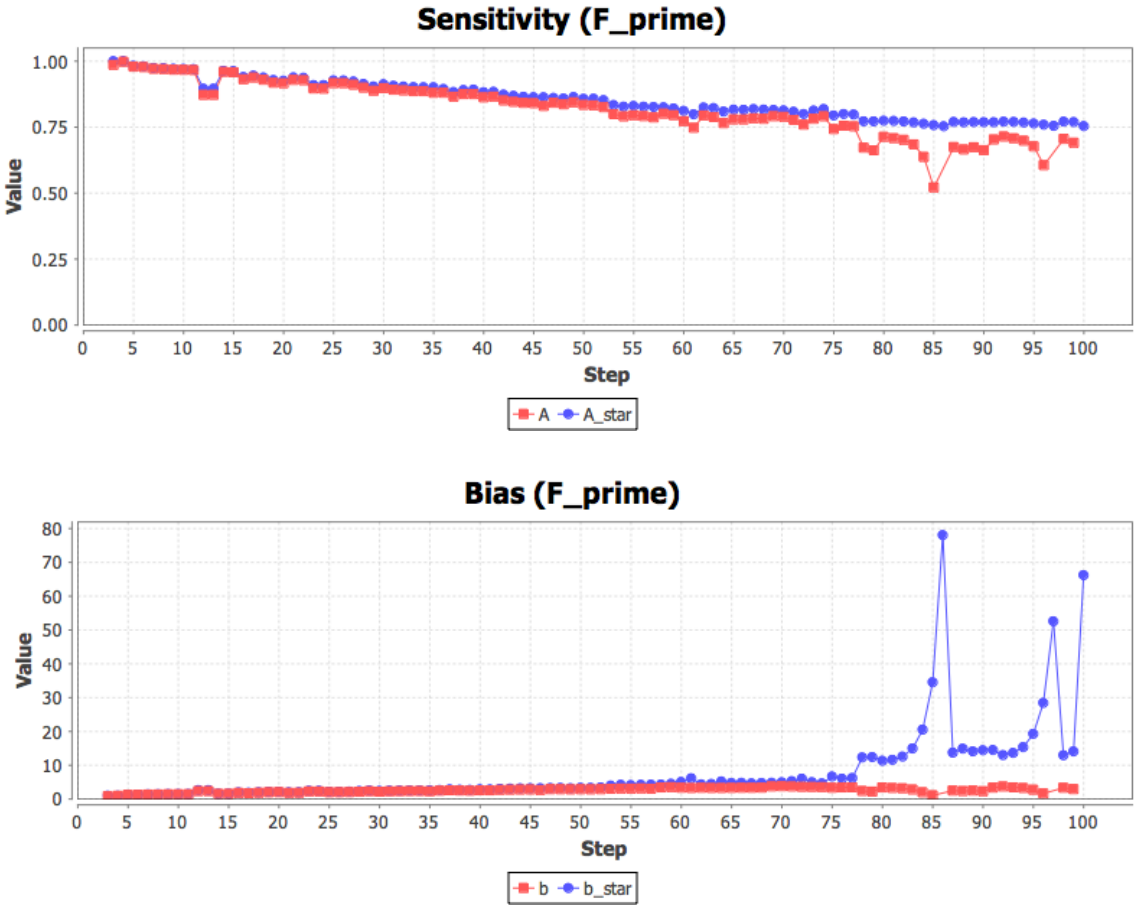

Figure D.5: *Sensitivity and bias using F' for graph "West1 A" and batch execution "East".*

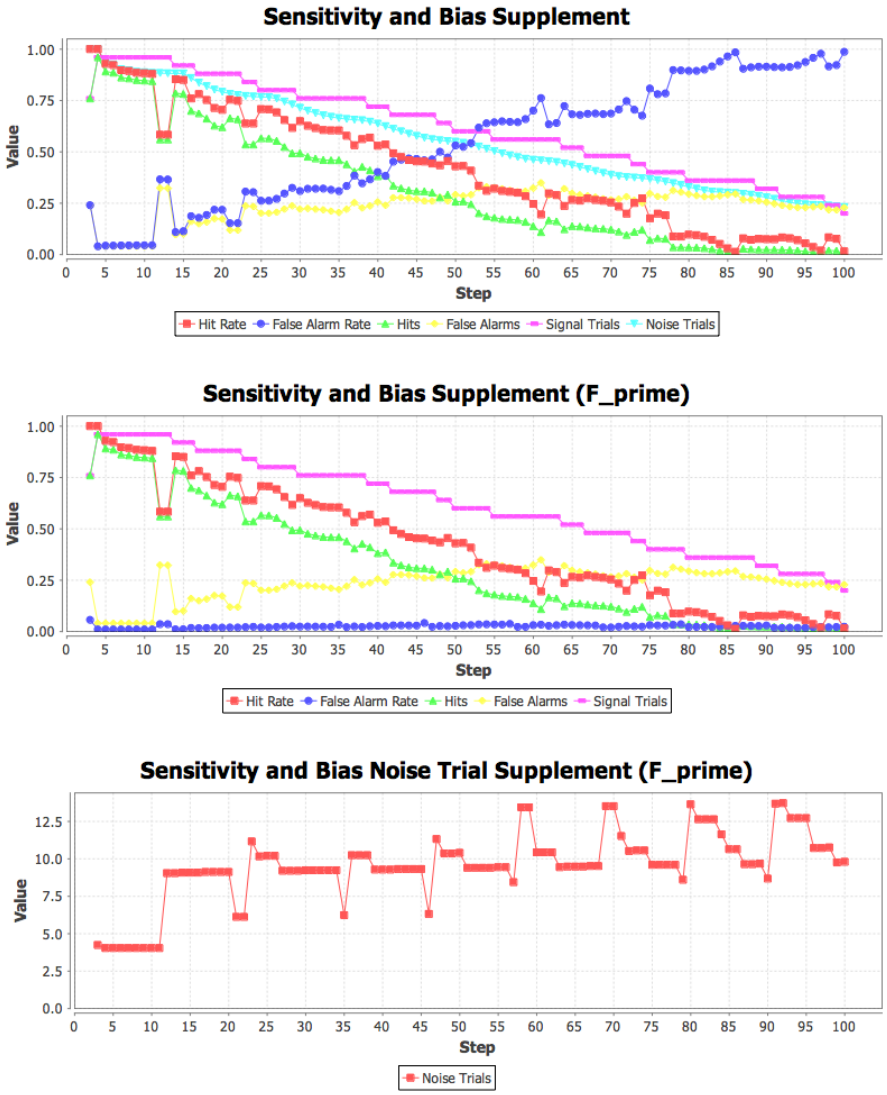

Figure D.6: *Additional sensitivity and bias plots for graph "West1 A" and batch execution "East".*

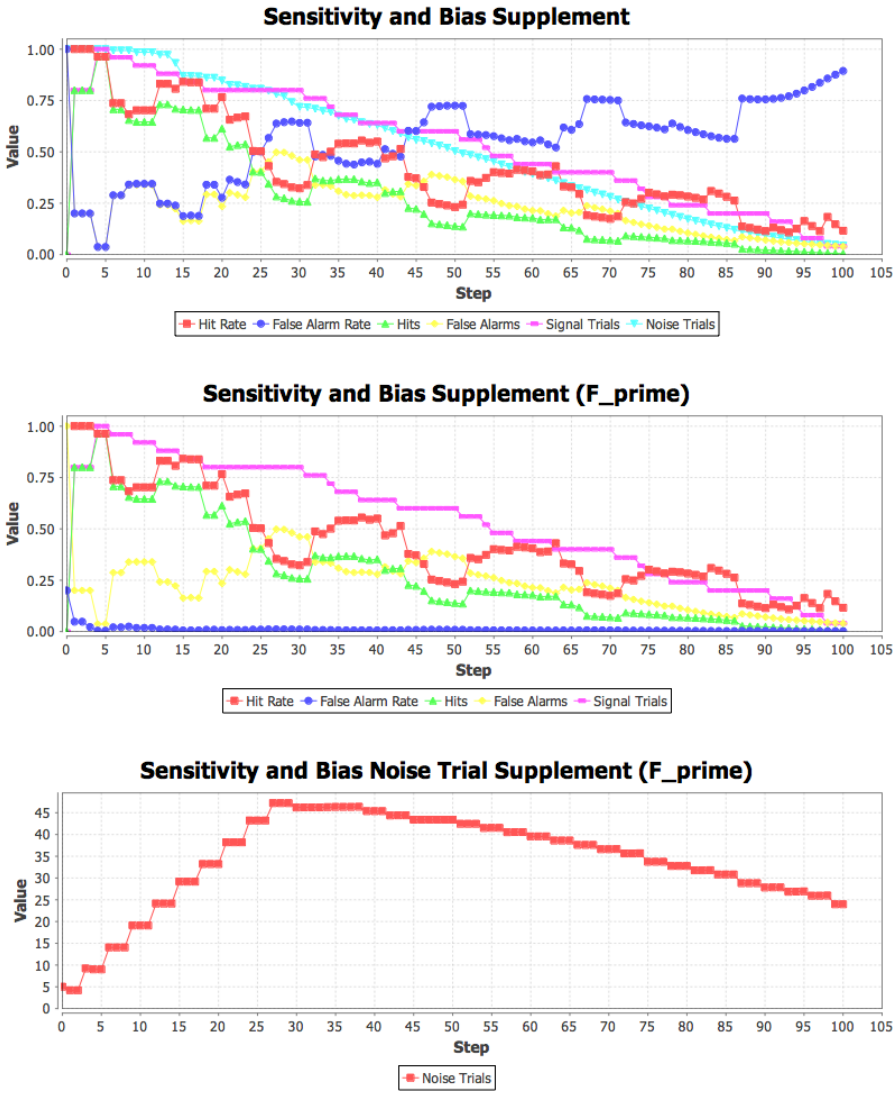

Figure D.7: *Additional sensitivity and bias plots for graph "West2 A" and batch execution "West2".*

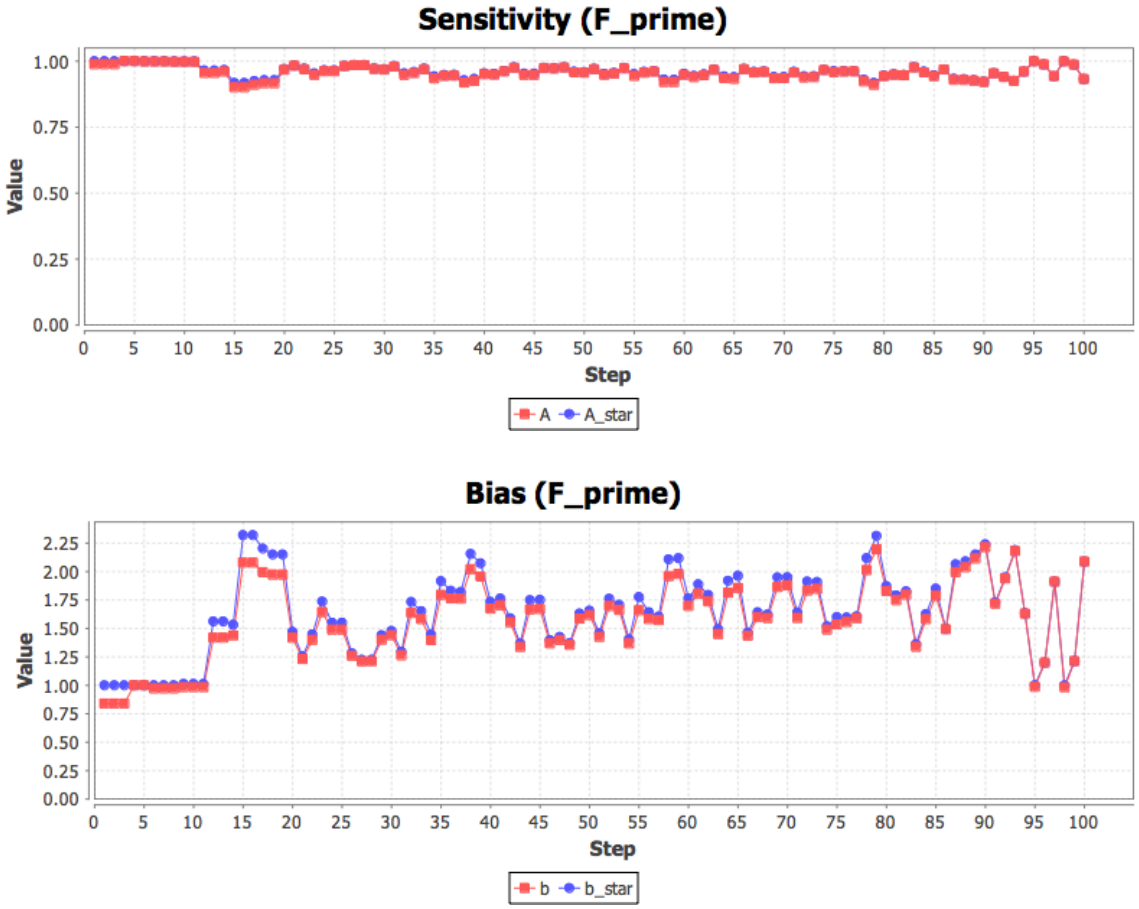

Figure D.8: *Sensitivity and bias using F' for graph "West2 B".*

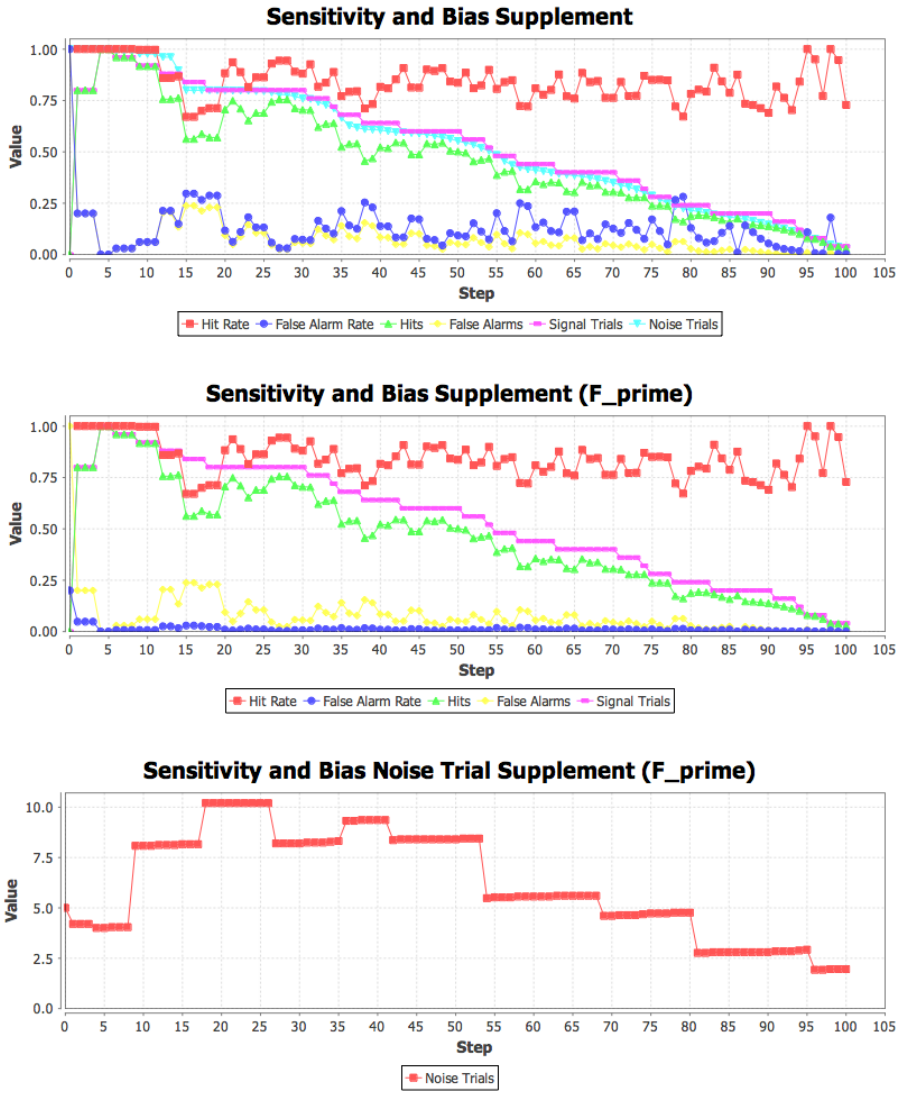

Figure D.9: *Additional sensitivity and bias plots for graph "West2 B" and batch execution "West2".*

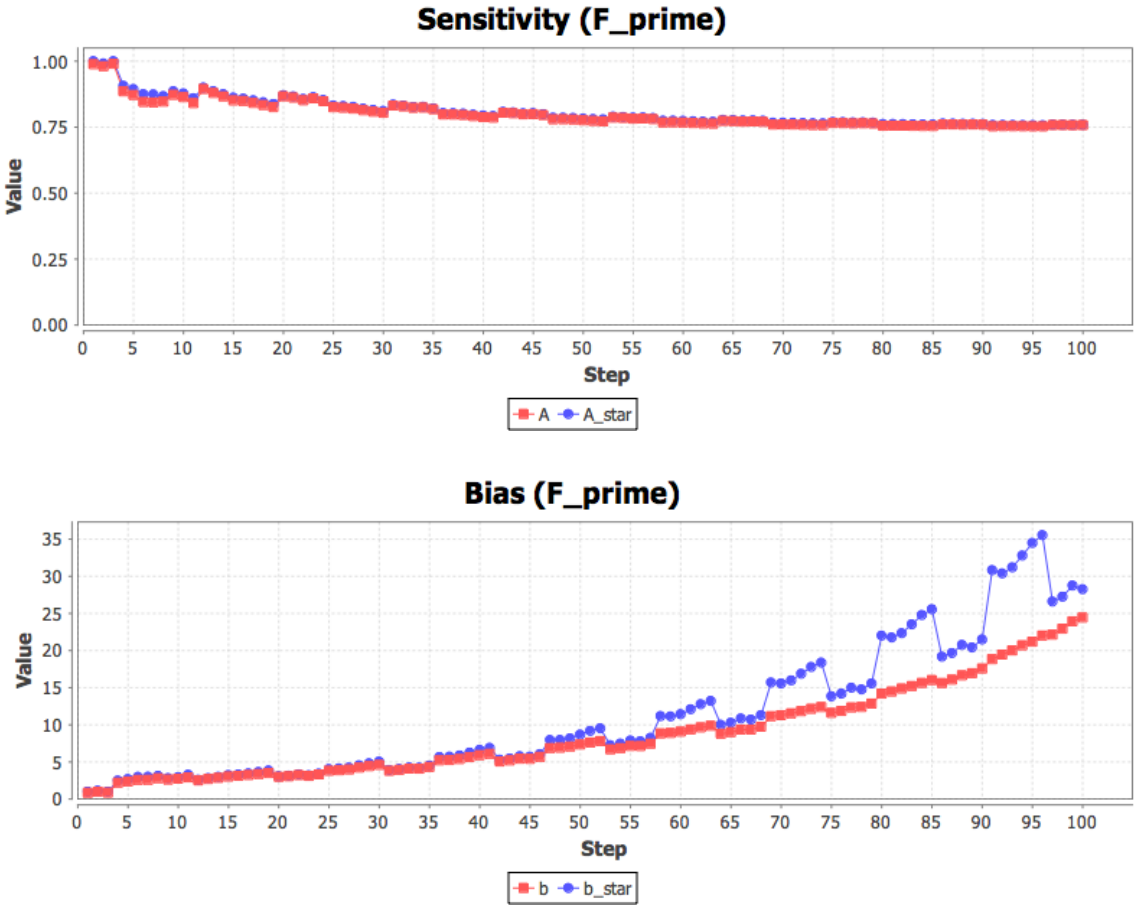

Figure D.10: *Sensitivity and bias using F' for graph "West3 A".*

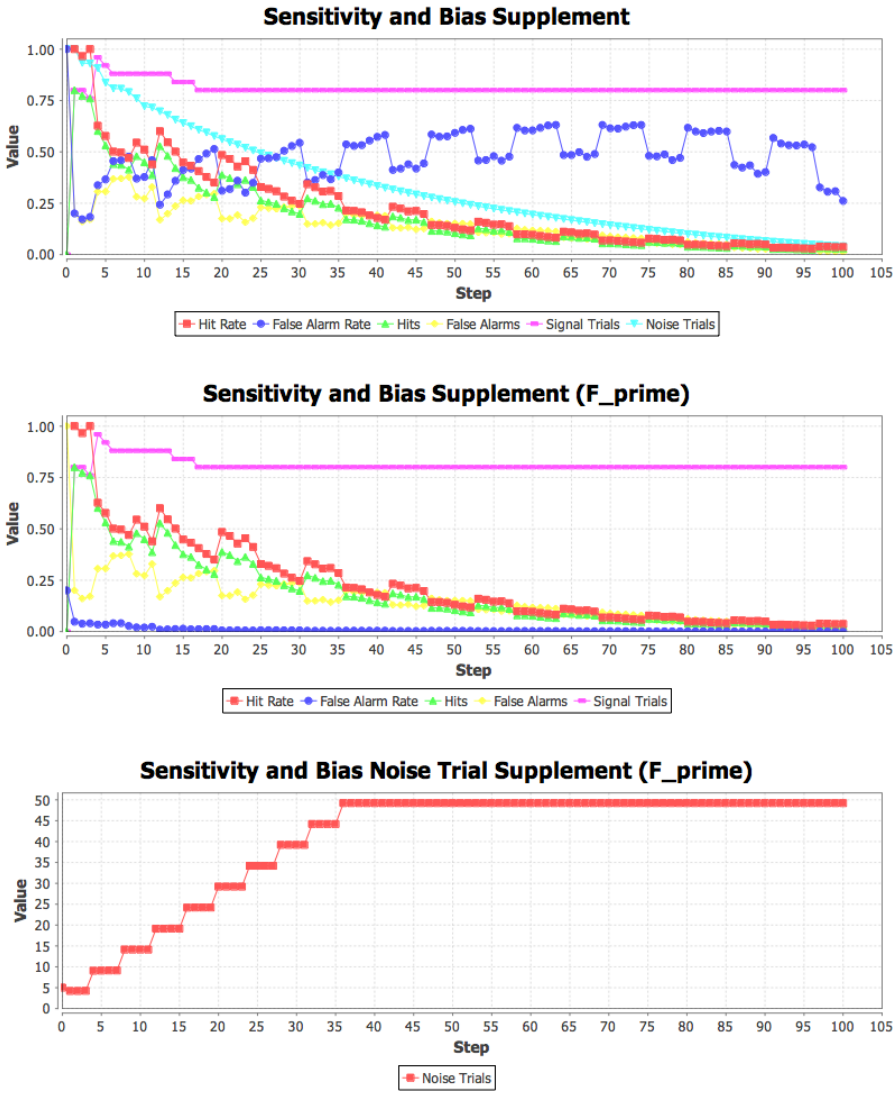

Figure D.11: *Additional sensitivity and bias plots for graph "West3 A" and batch execution "West3".*

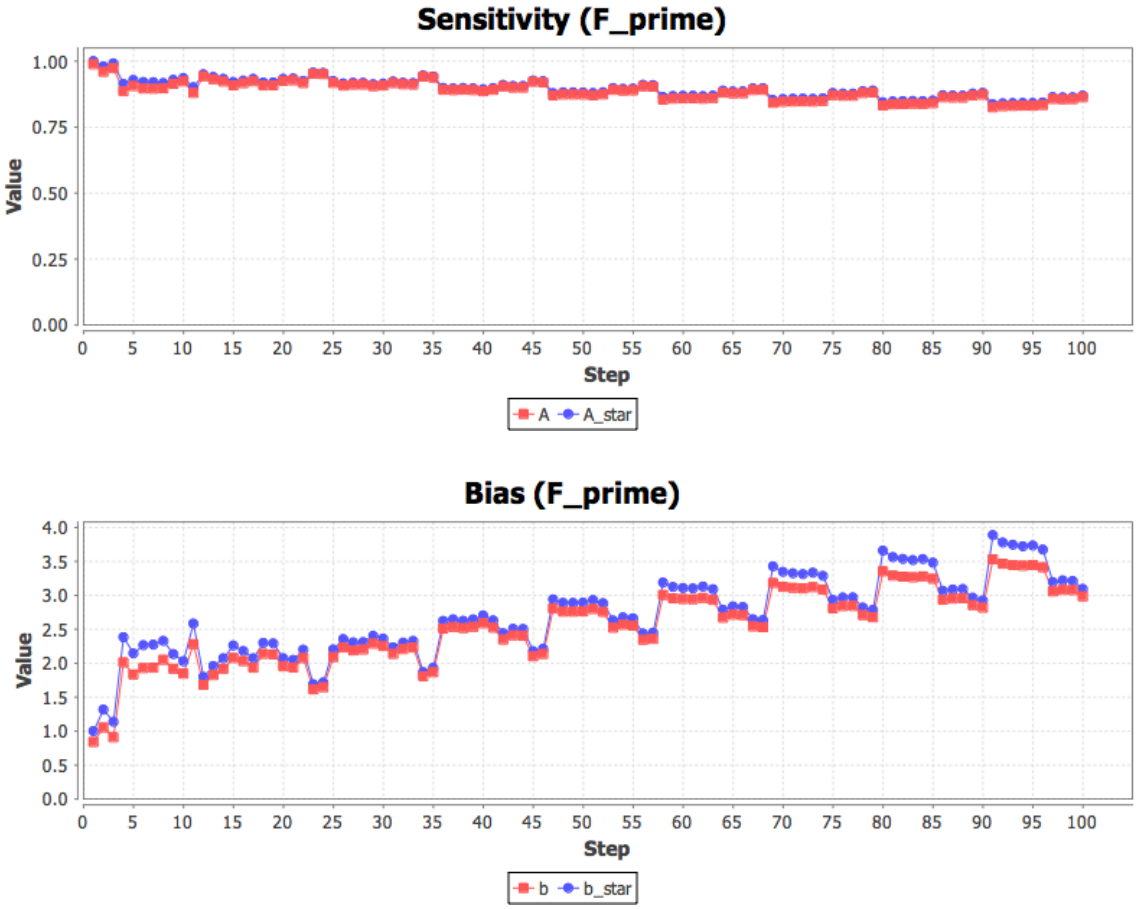

Figure D.12: *Sensitivity and bias using F' for graph "West3 B".*

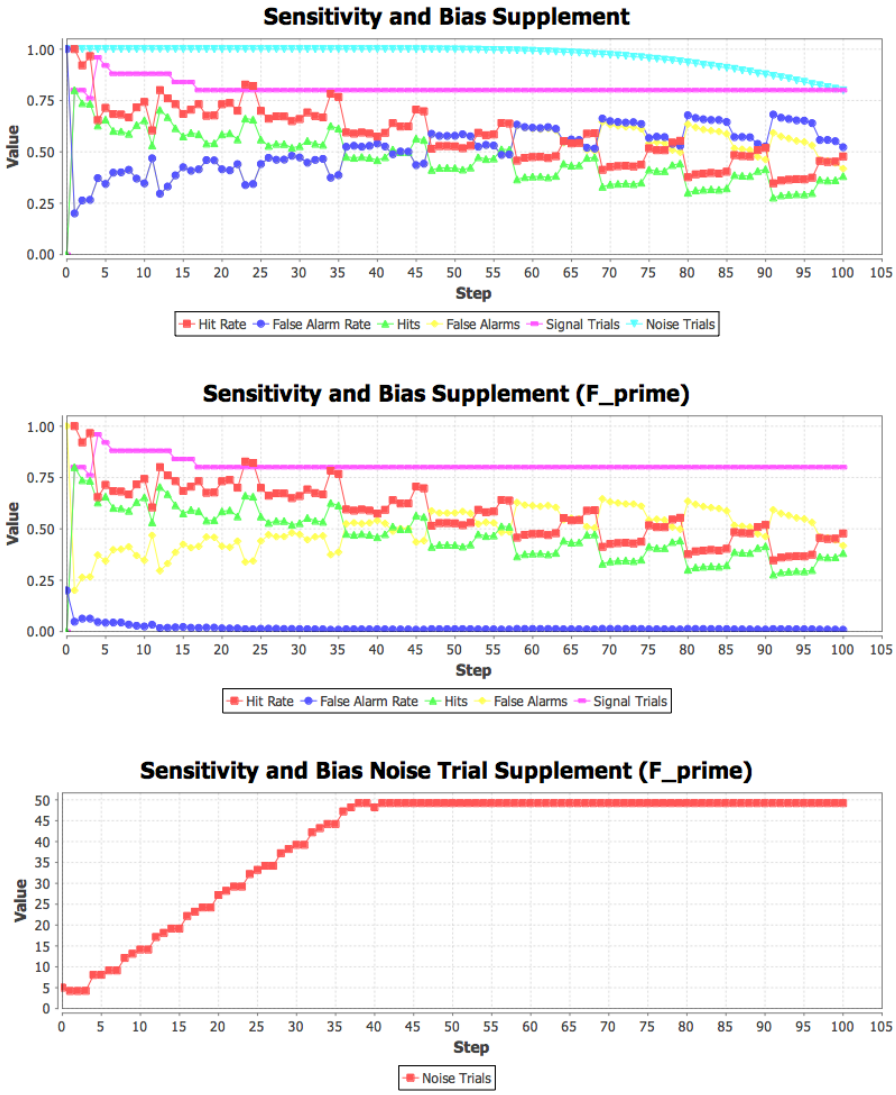

Figure D.13: *Additional sensitivity and bias plots for graph "West3 B" and batch execution "West3".*

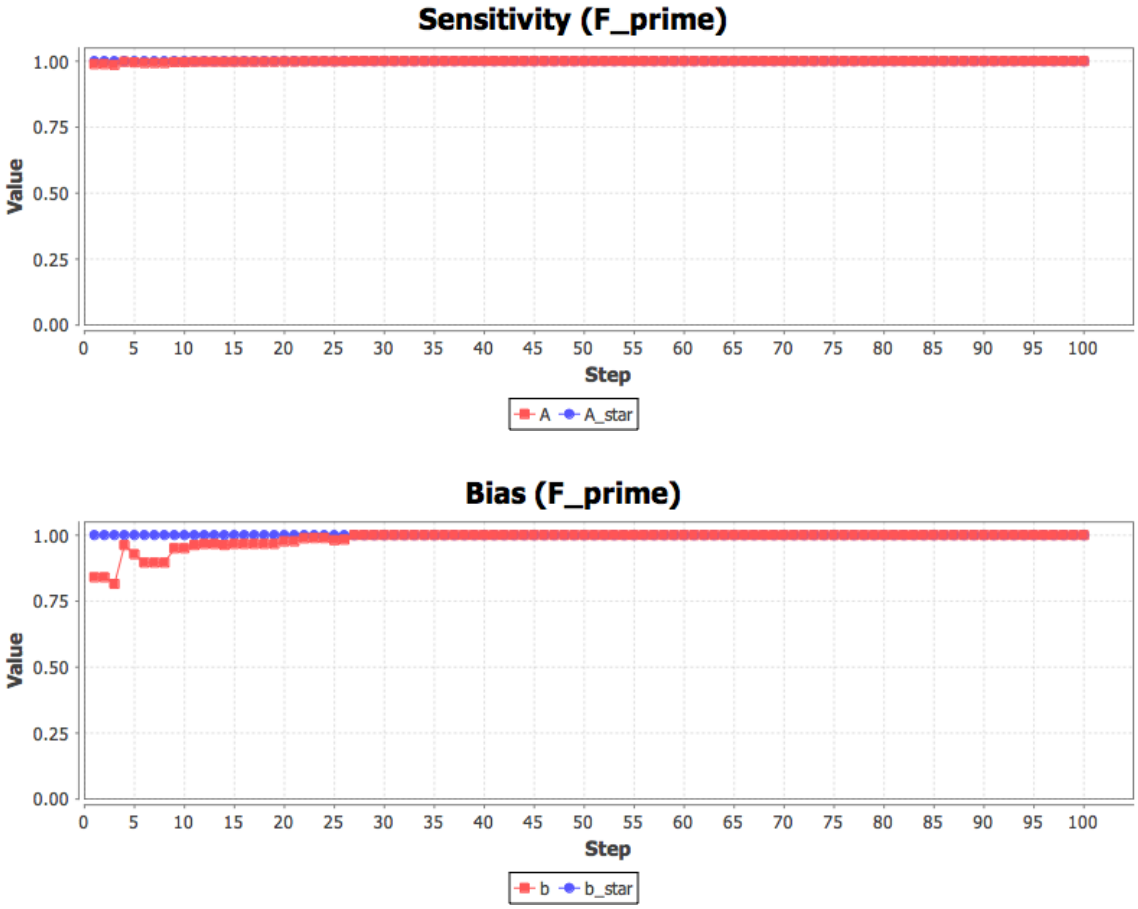

Figure D.14: *Sensitivity and bias using F' for graph "West3 C".*

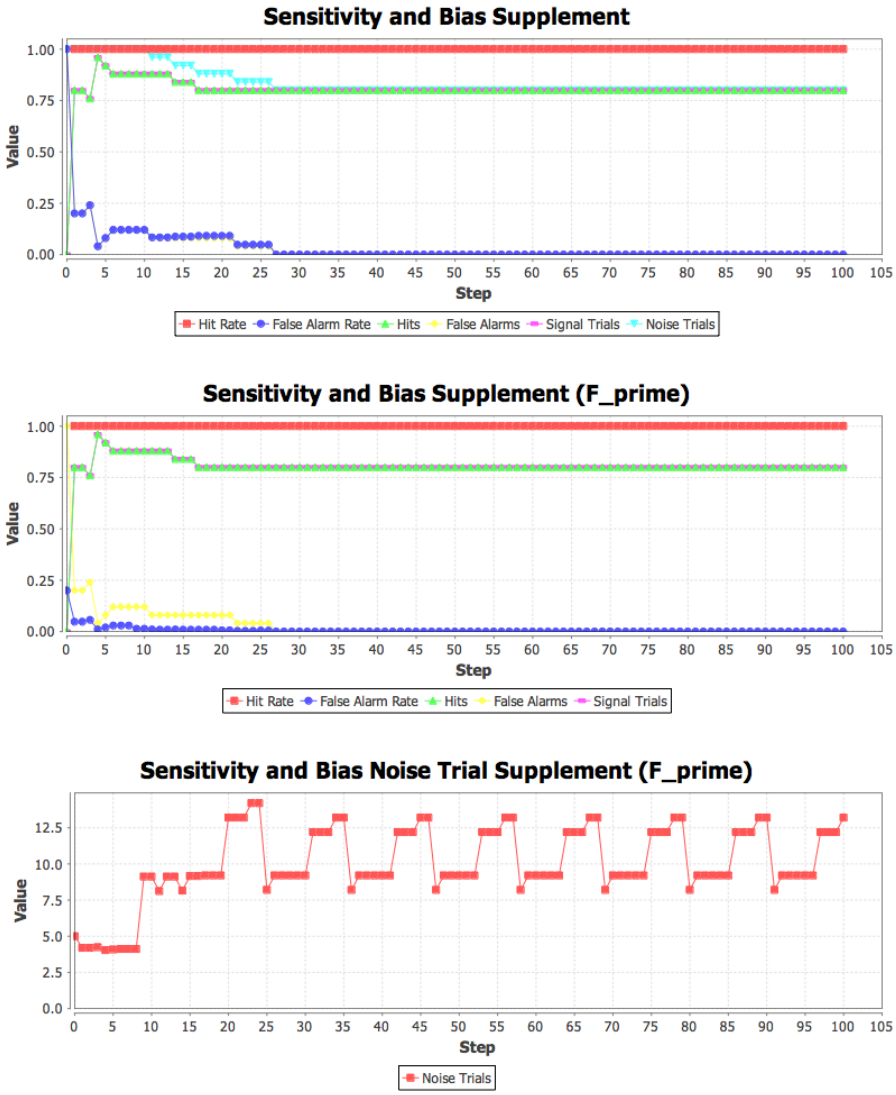

Figure D.15: *Additional sensitivity and bias plots for graph "West3 C" and batch execution "West3".*

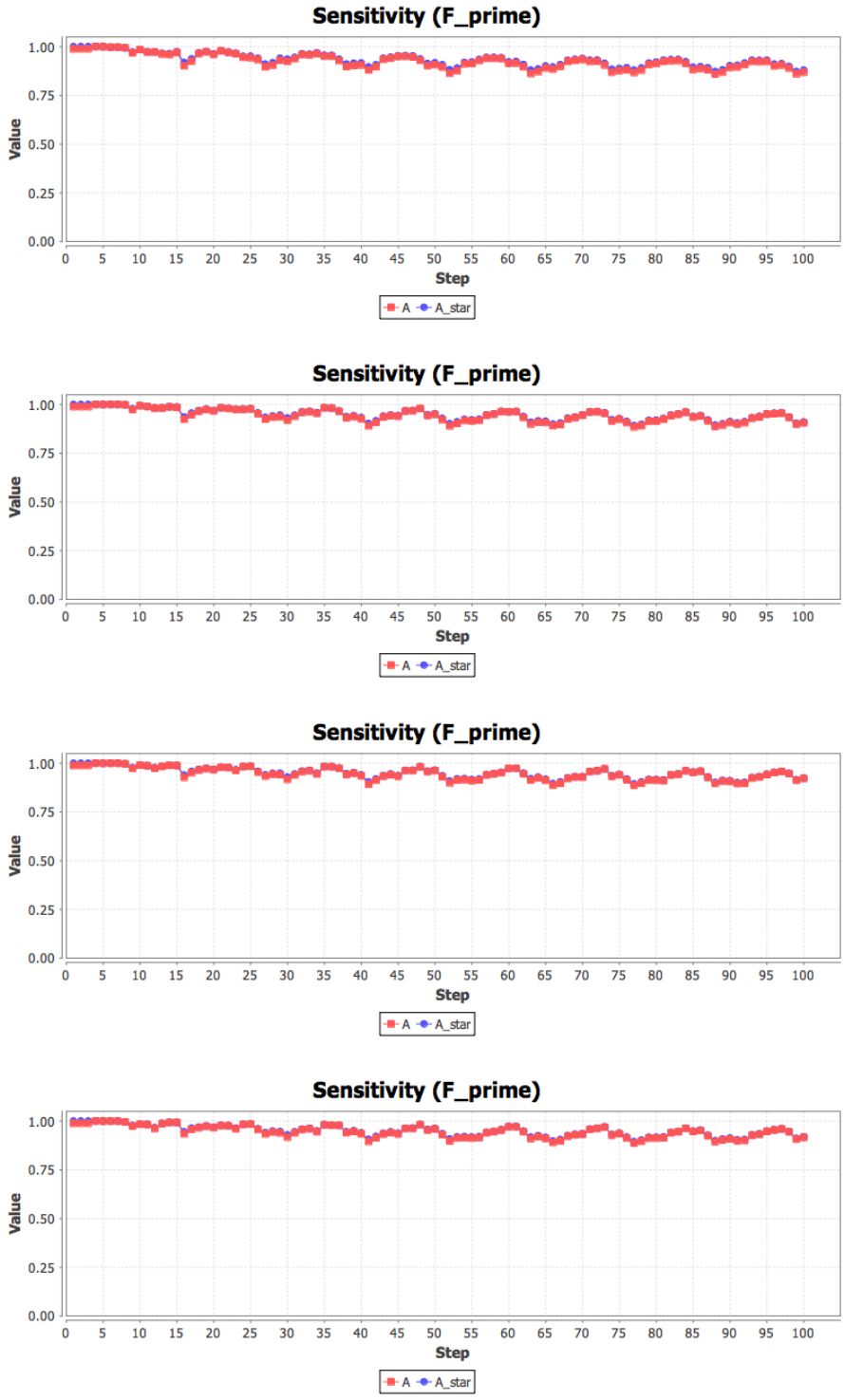

Figure D.16: *Sensitivity and bias using F' for graph "West4".*

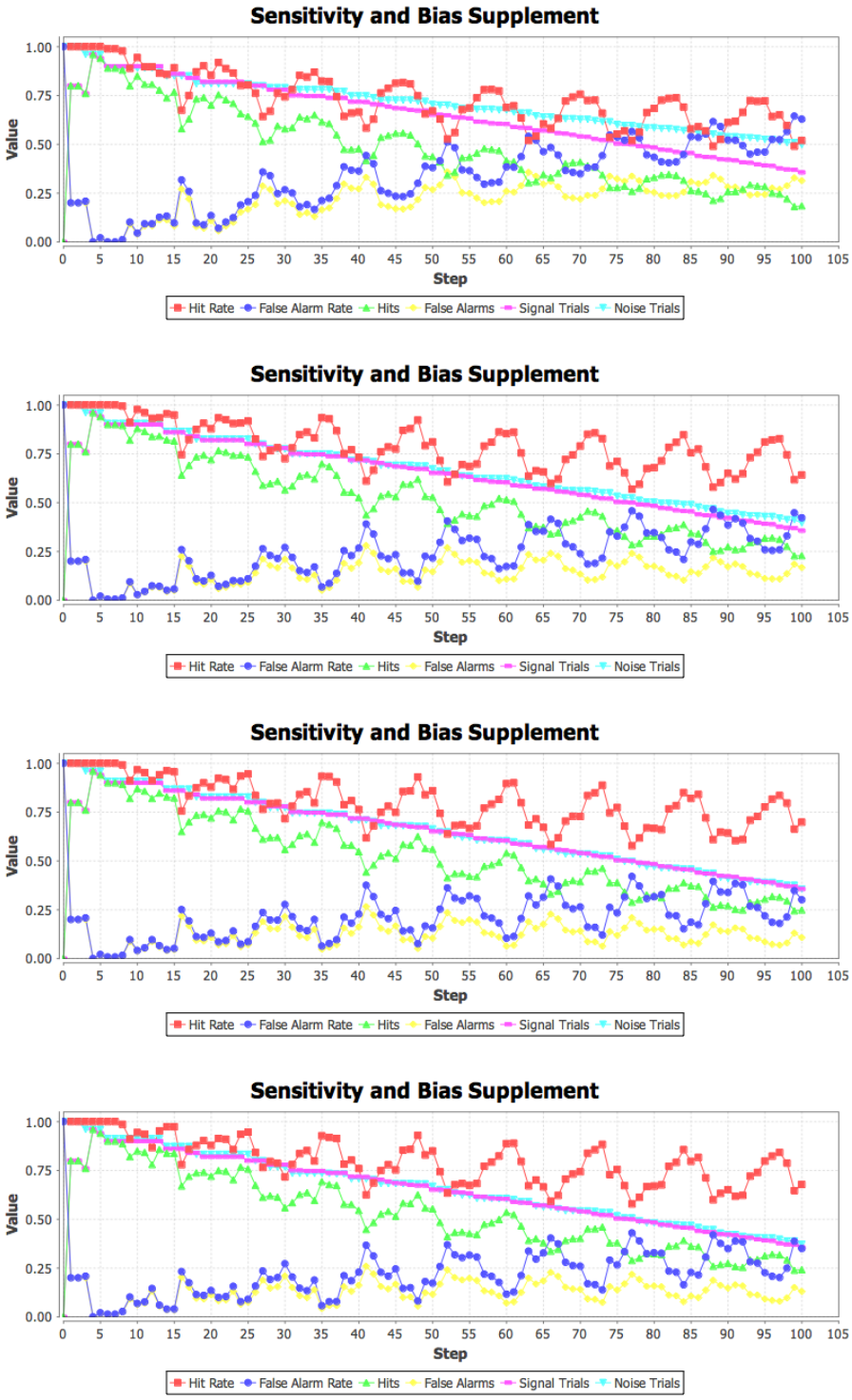

Figure D.17: *Additional sensitivity and bias plots for graph "West4" and batch execution "West4".*

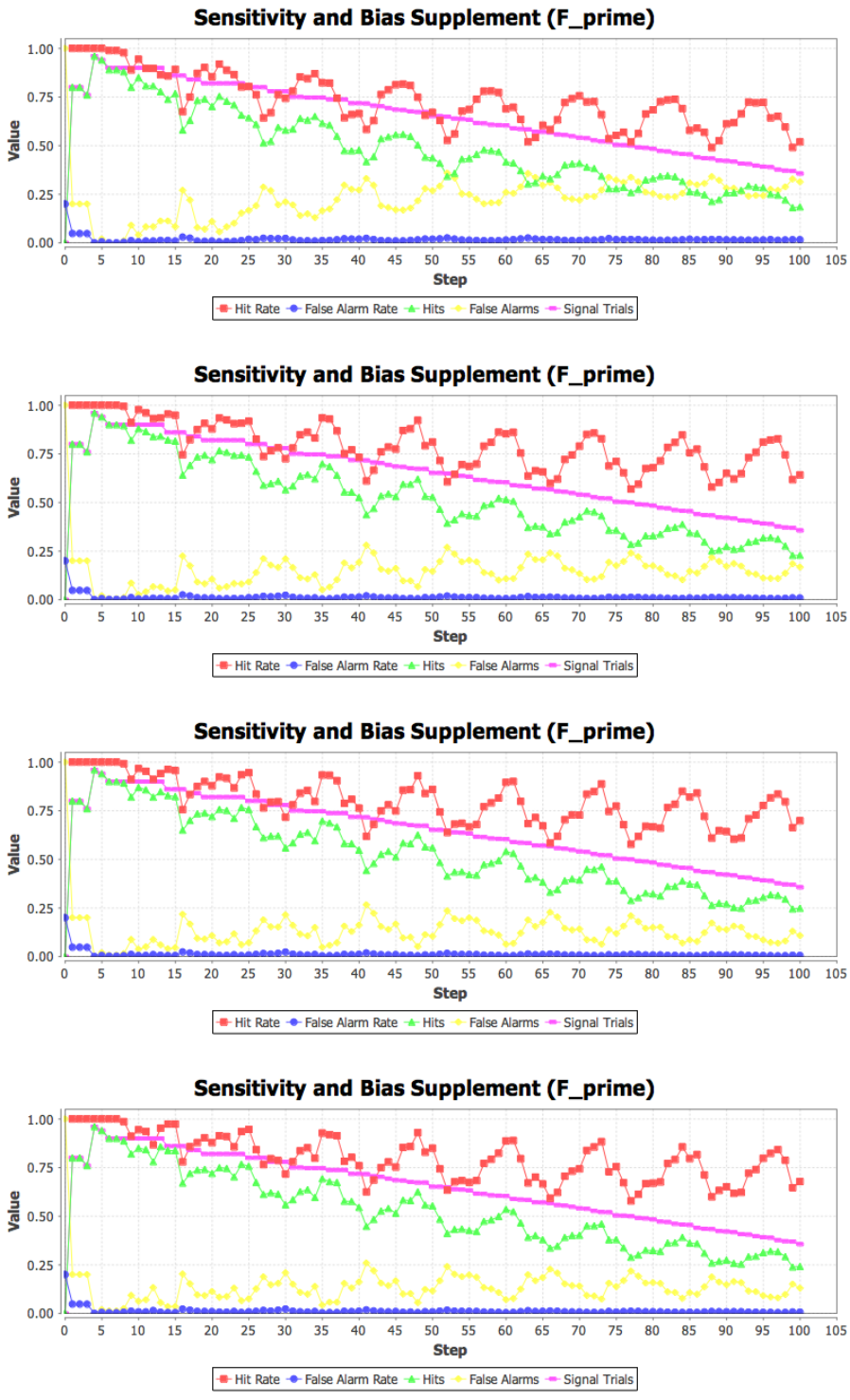

Figure D.18: *Additional sensitivity and bias plots for graph "West4" and batch execution "West4".*

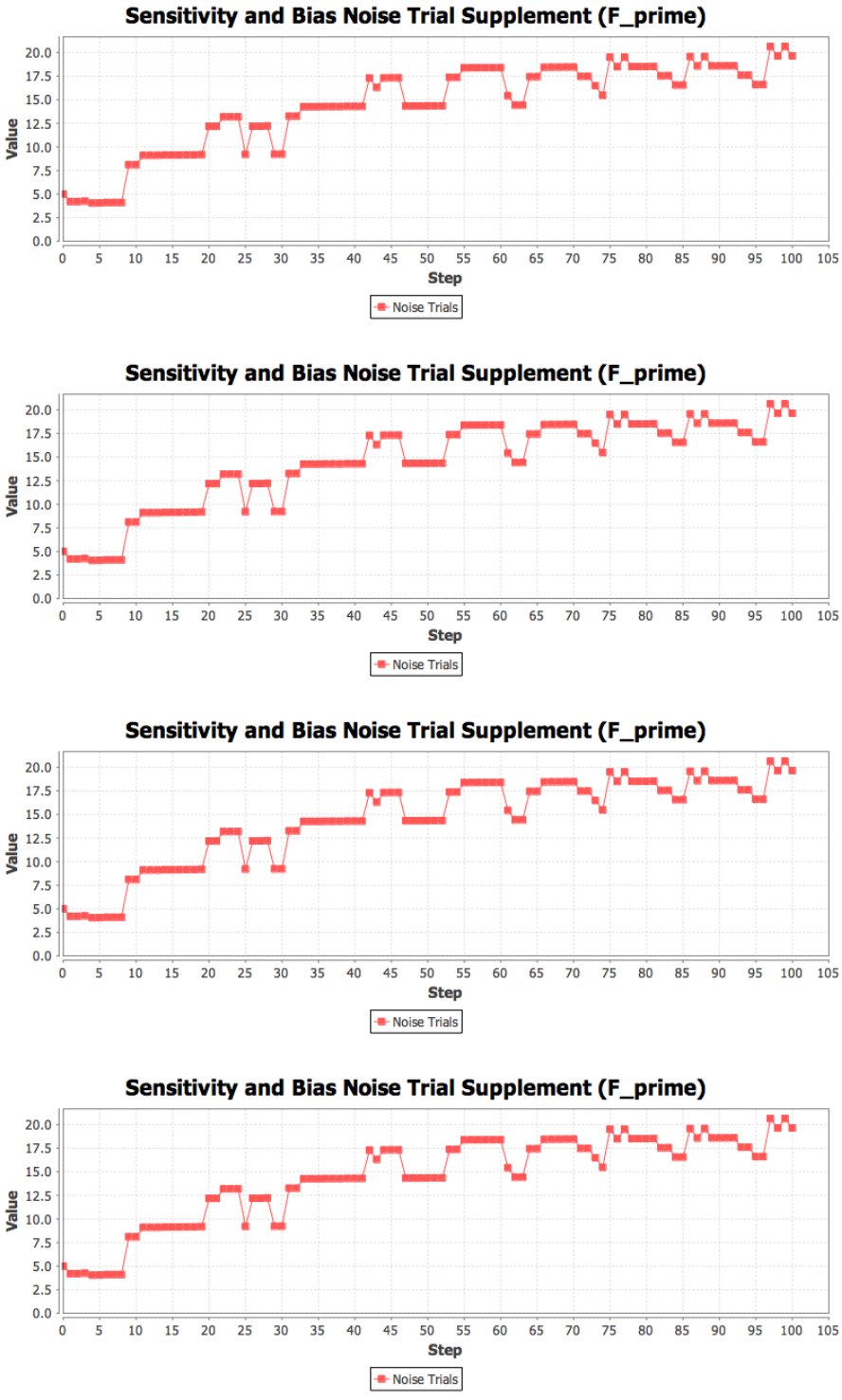

Figure D.19: *Additional sensitivity and bias plots for graph "West4" and batch execution "West4".*

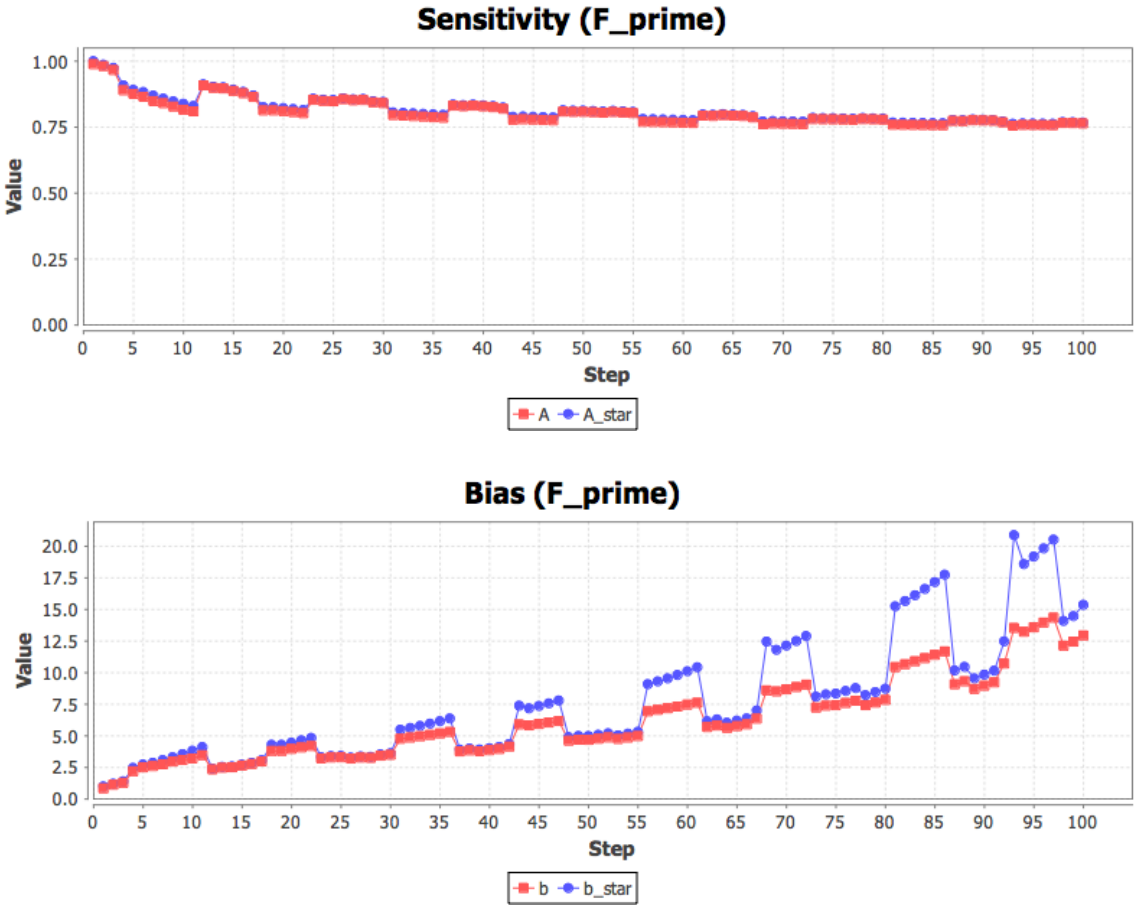

Figure D.20: *Sensitivity and bias using F' for graph "West1 B".*

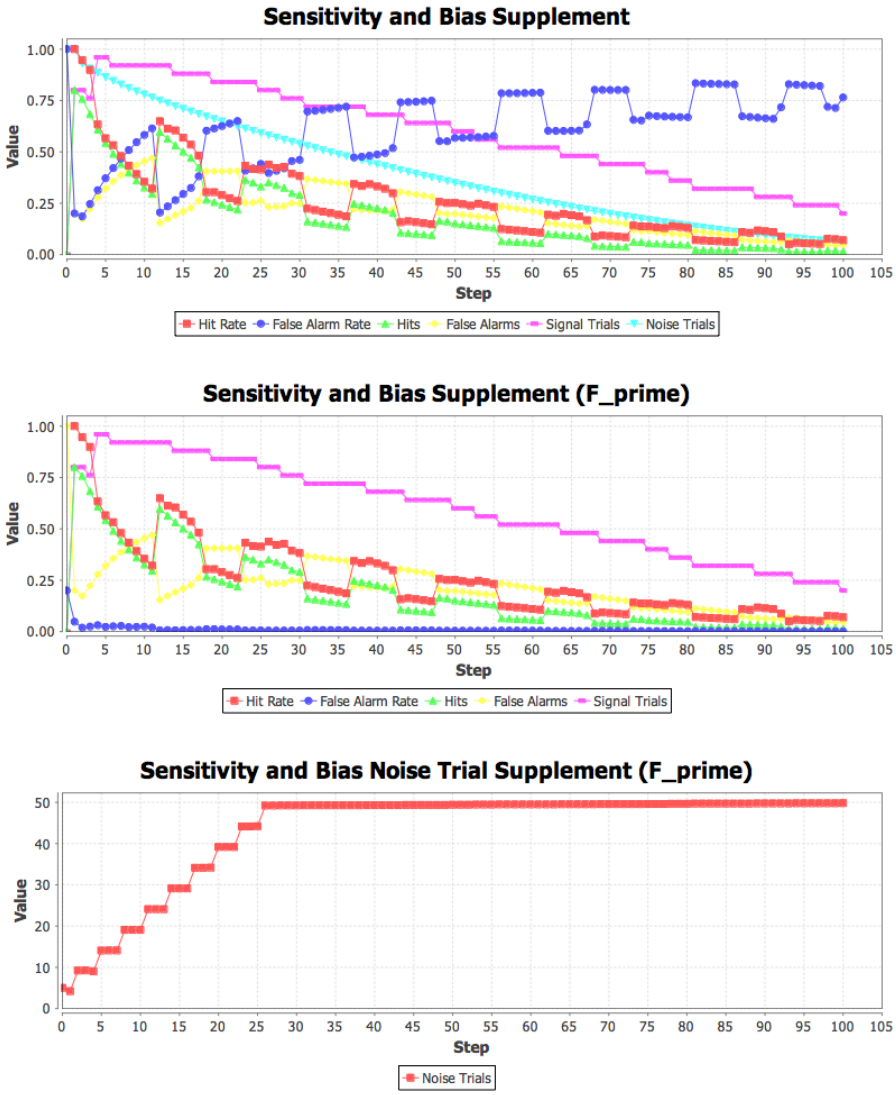

Figure D.21: *Additional sensitivity and bias plots for graph "West1 B" and batch execution "West1".*

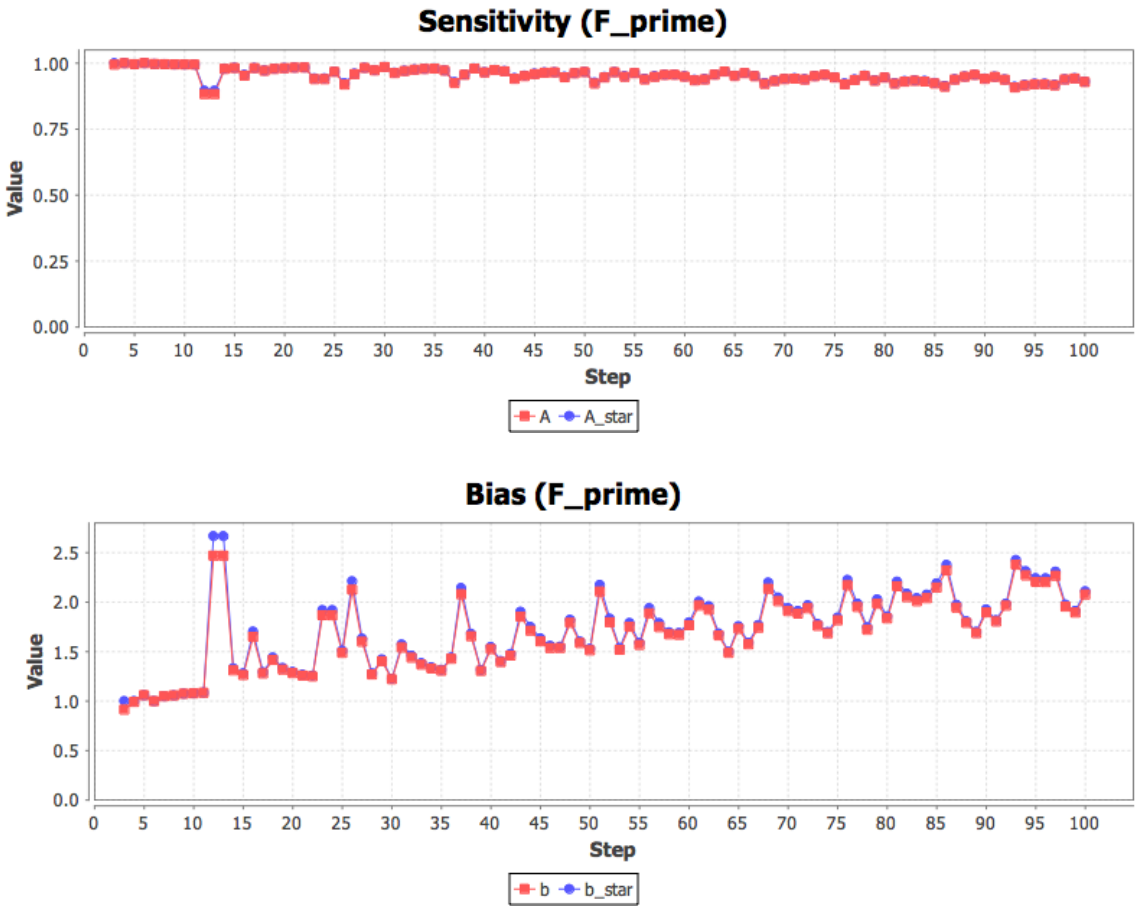

Figure D.22: *Sensitivity and bias using F' for graph "West1 A" for batch execution "West1 Large".*

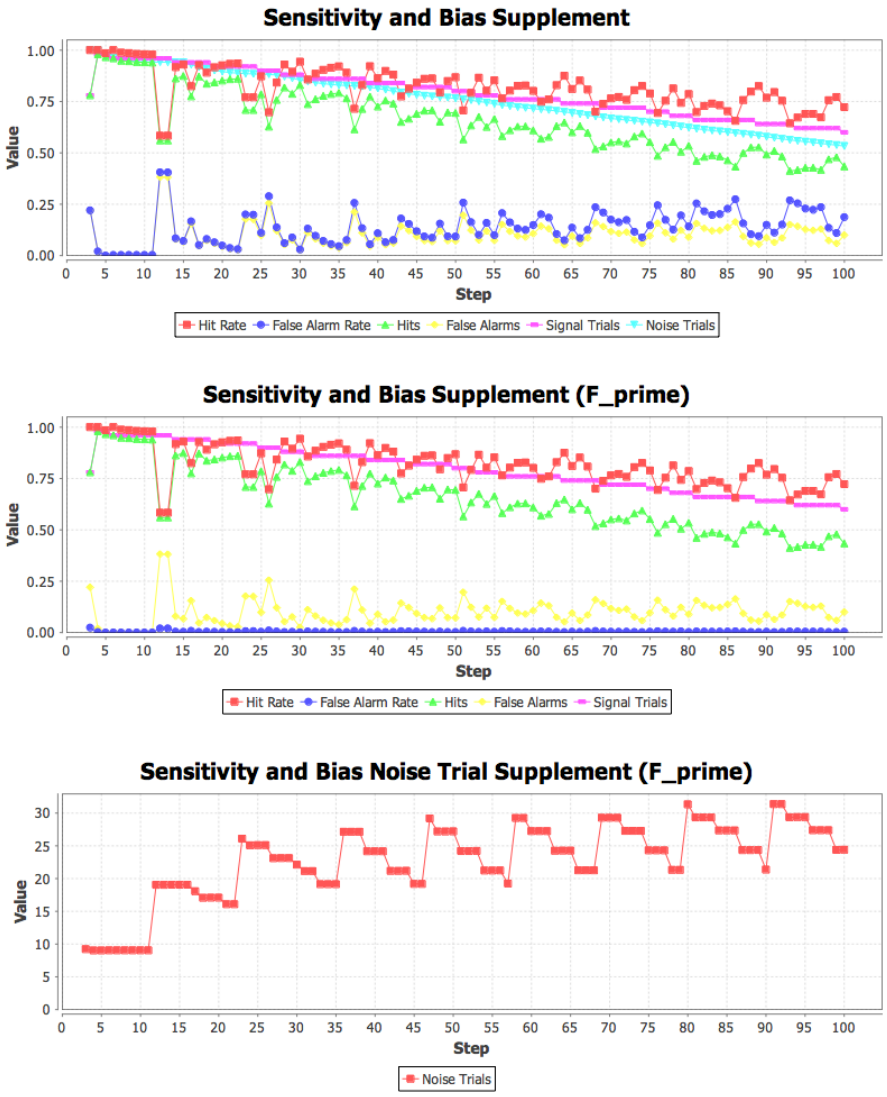

Figure D.23: *Additional sensitivity and bias plots for graph "West1 A" and batch execution "West1 Large".*

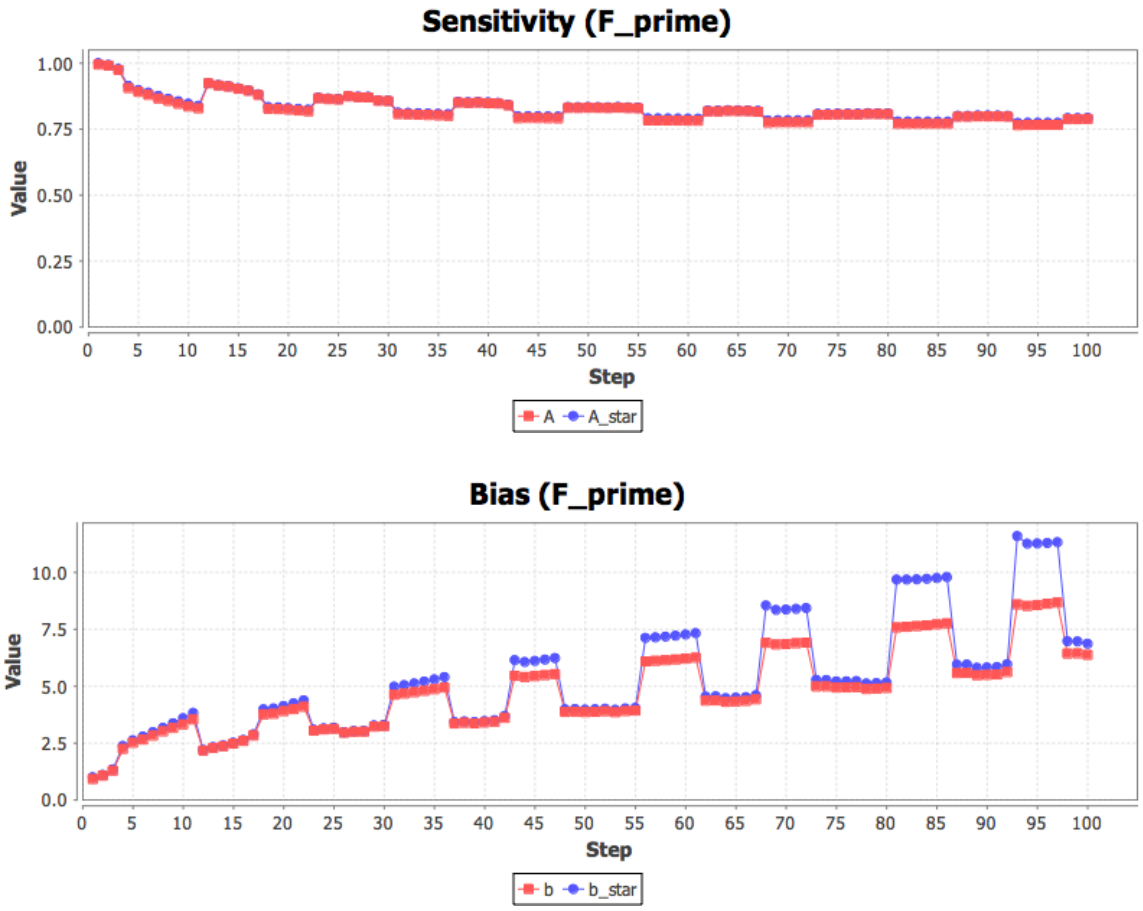

Figure D.24: *Sensitivity and bias using F' for graph "West1 B" for batch execution "West1 Large".*

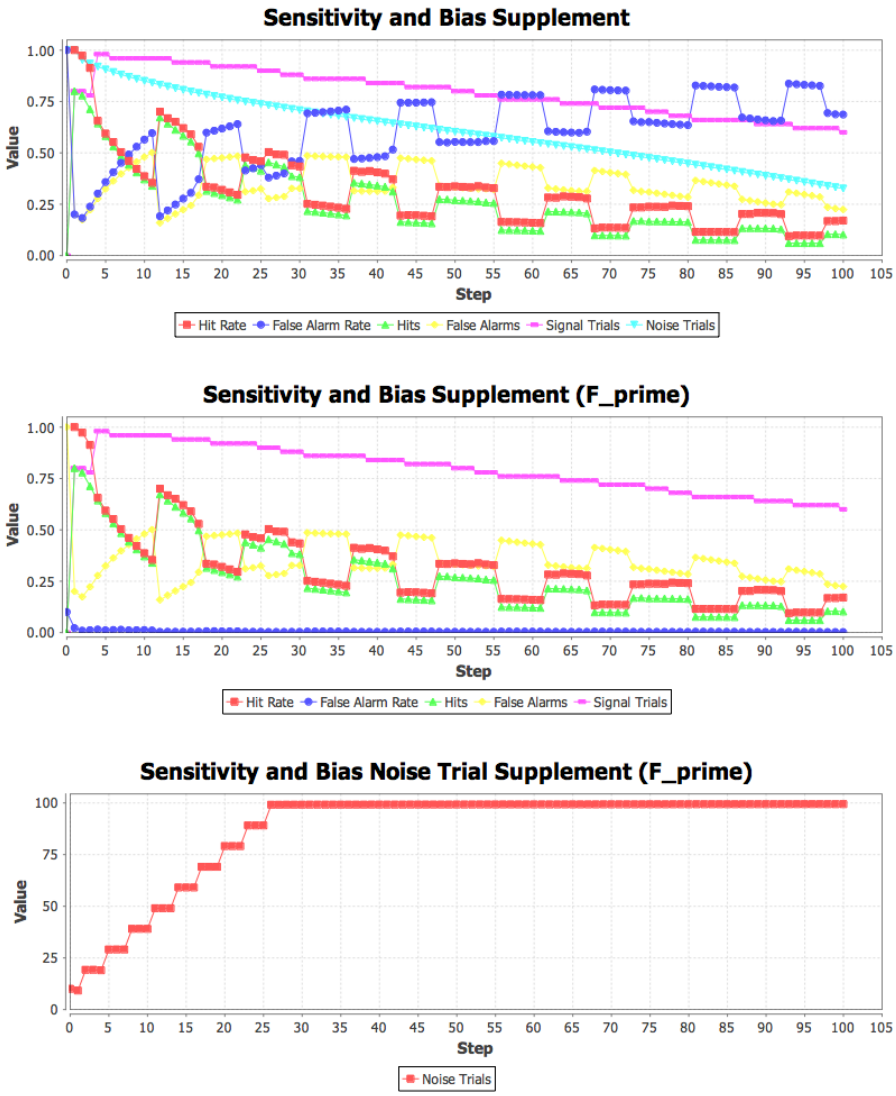

Figure D.25: *Additional sensitivity and bias plots for graph "West1 B" and batch execution "West1 Large".*

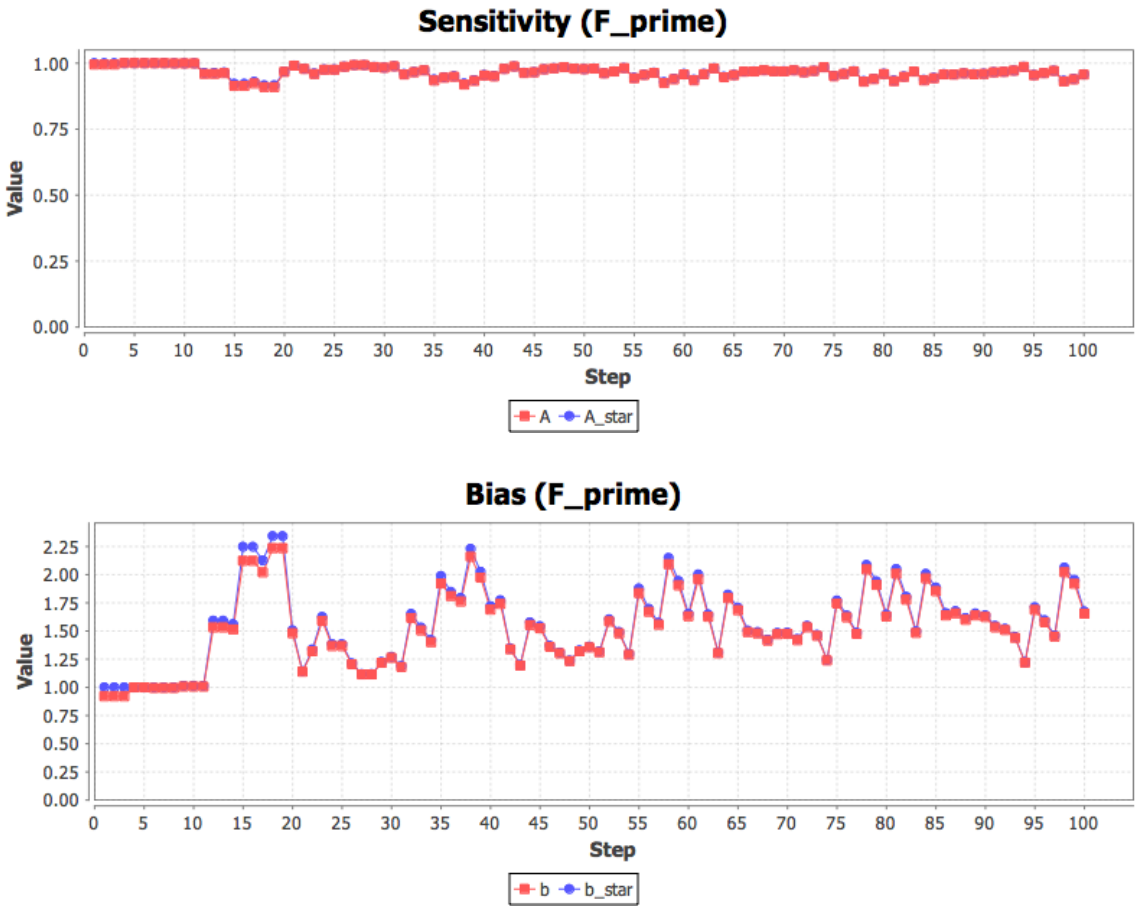

Figure D.26: *Sensitivity and bias using F' for graph "West2 B" for batch execution "West2 Large".*

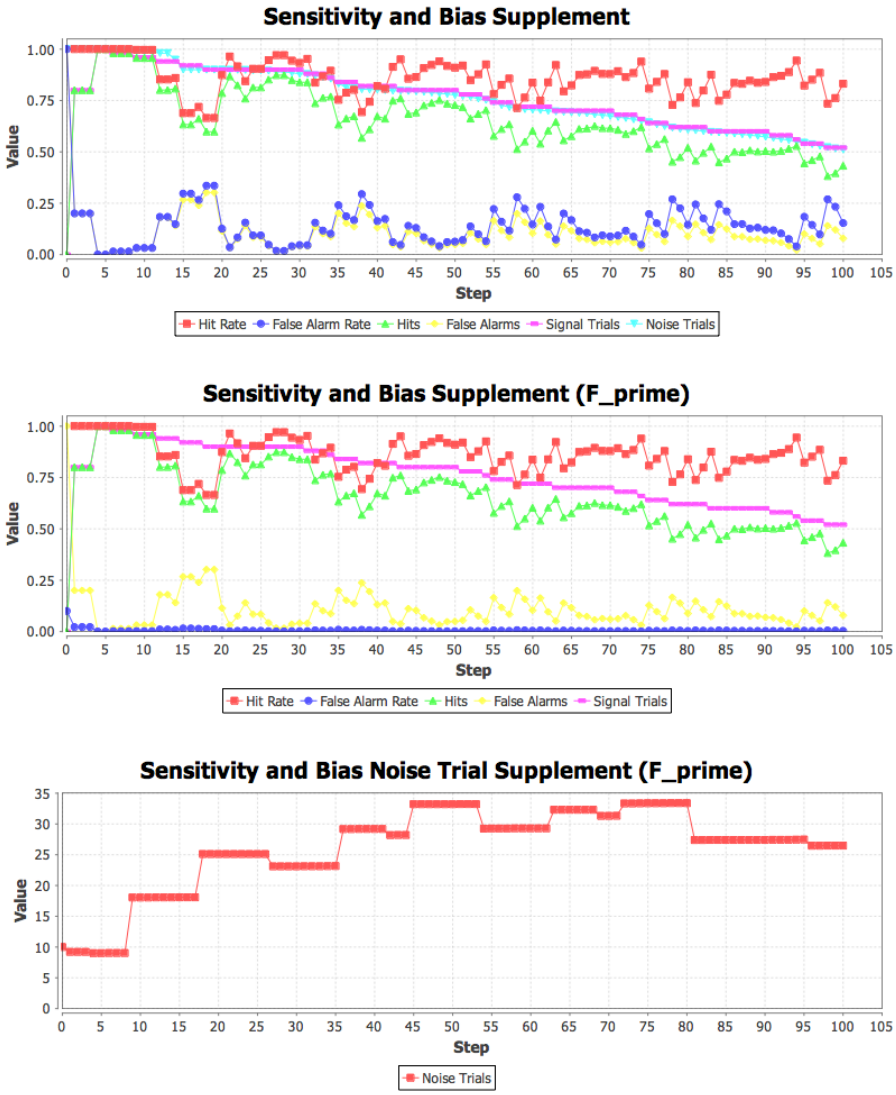

Figure D.27: *Additional sensitivity and bias plots for graph "West2 B" and batch execution "West2".*

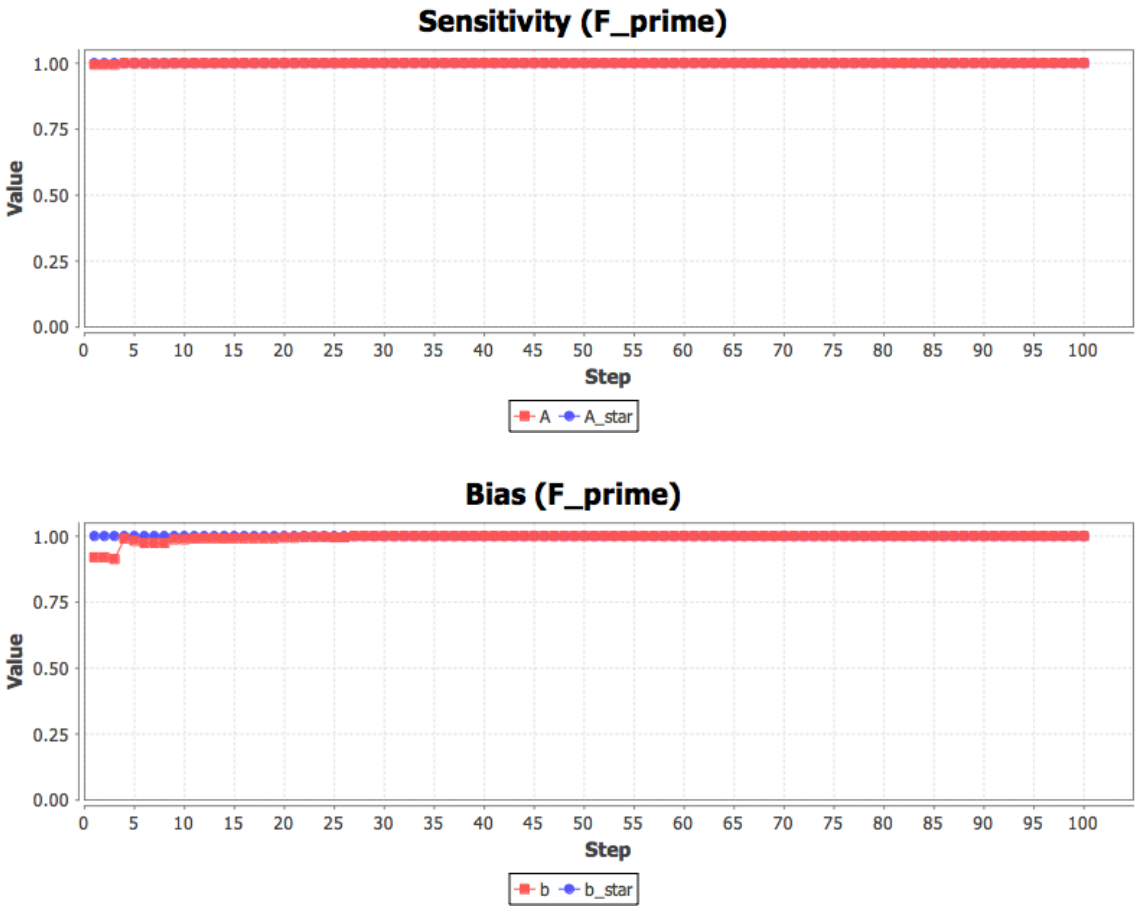

Figure D.28: *Sensitivity and bias using F' for graph "West3 C" and batch execution "West3 Large".*

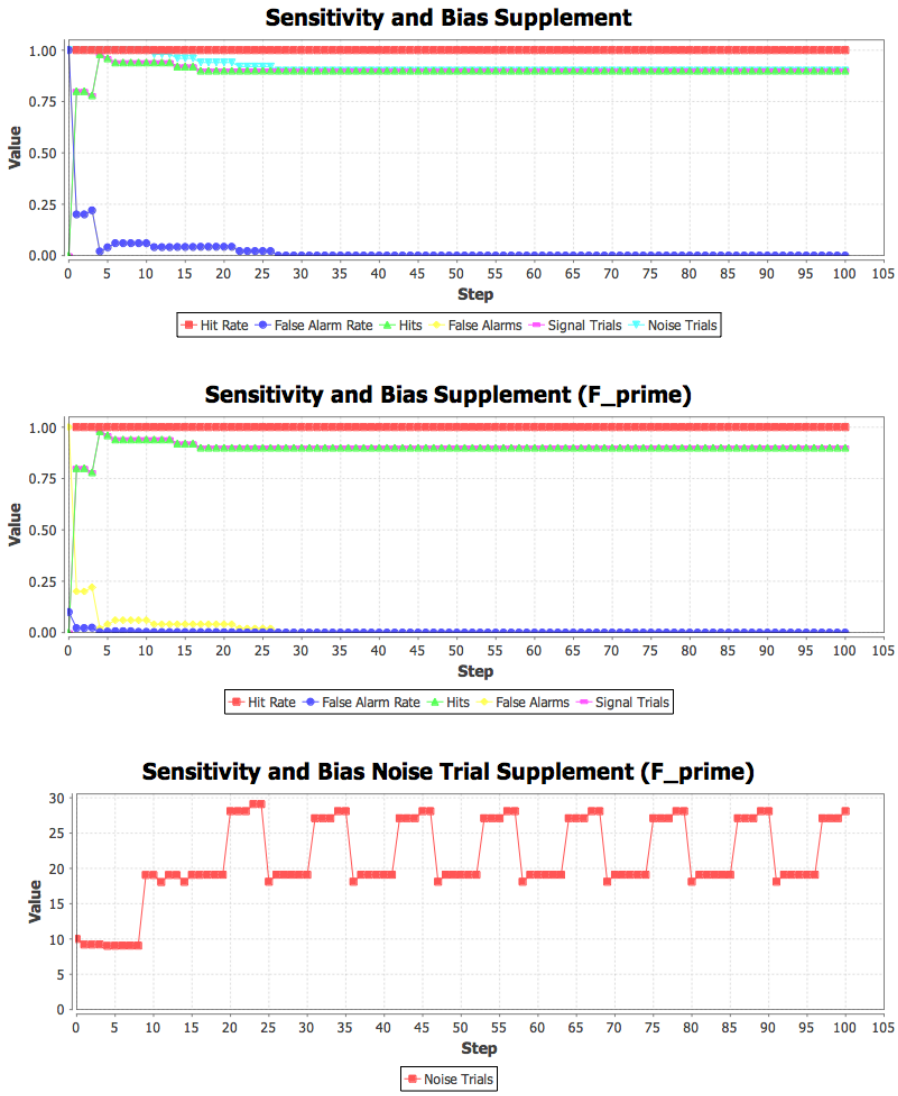

Figure D.29: *Additional sensitivity and bias plots for graph "West3 C" and batch execution "West3 Large".*

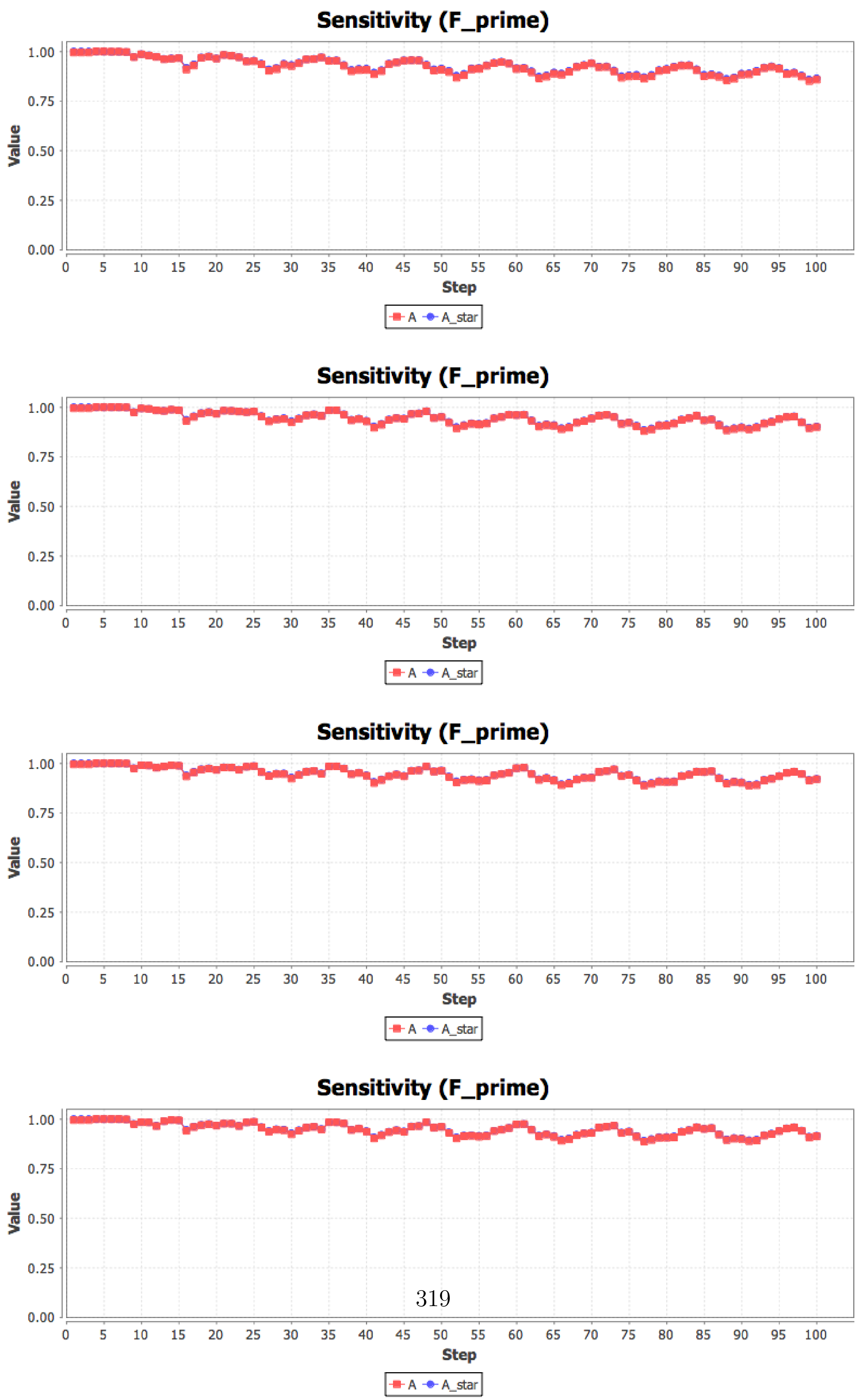

Figure D.30: *Sensitivity and bias using F' for graph "West4 C" for batch execution "West4*

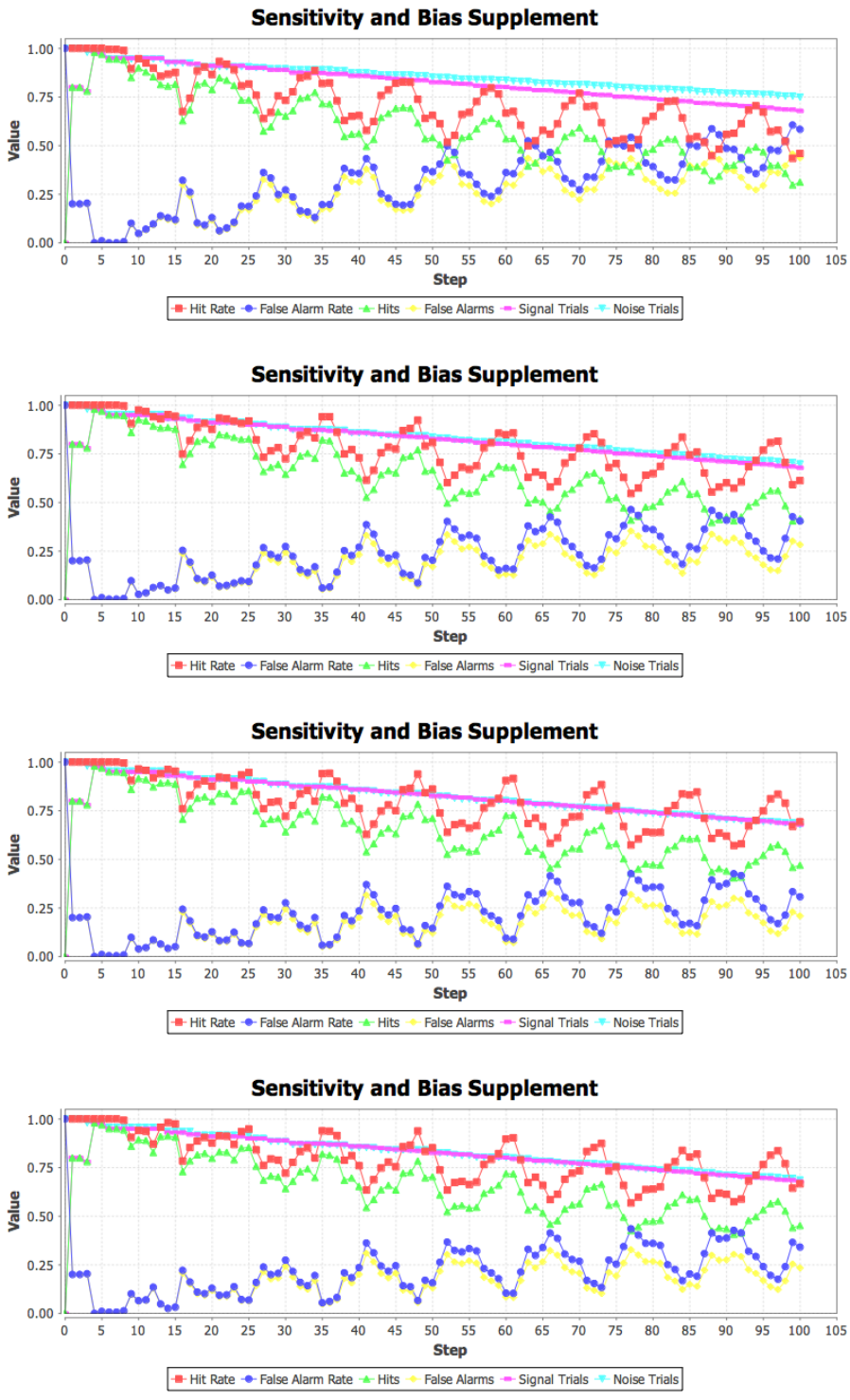

Figure D.31: *Additional sensitivity and bias plots for graph "West4" and batch execution "West4 Large".*

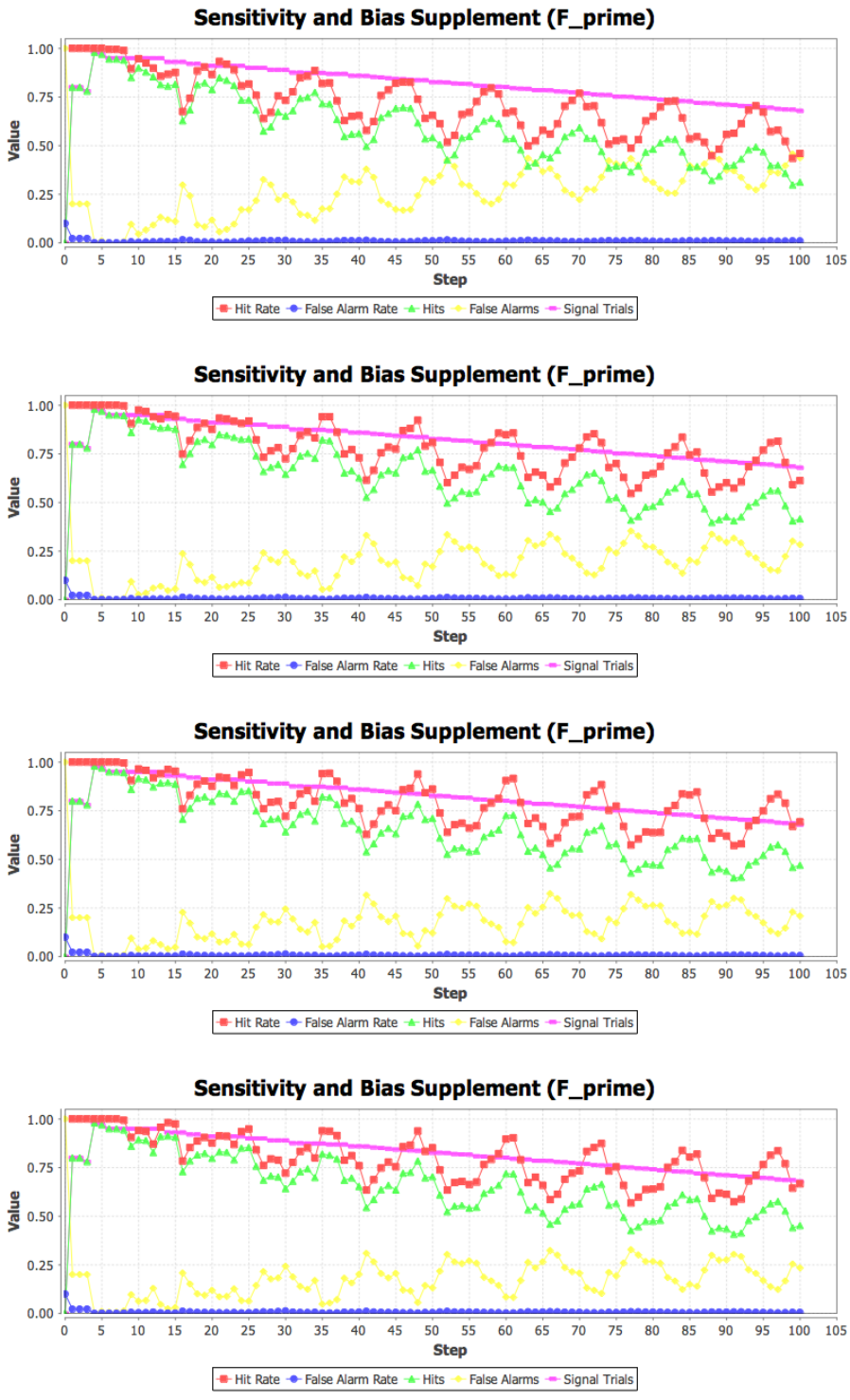

Figure D.32: *Additional sensitivity and bias plots for graph "West4" and batch execution "West4 Large".*

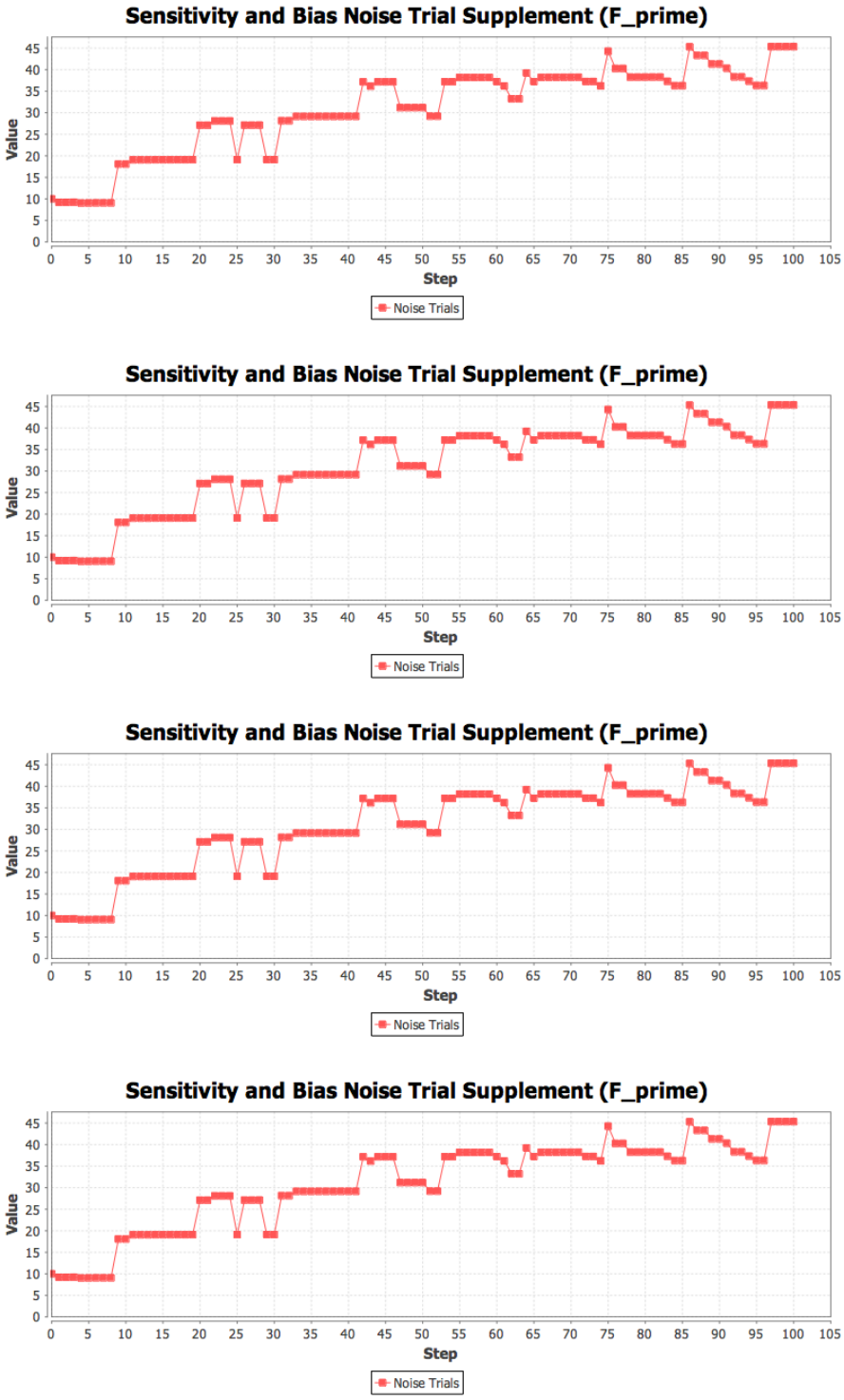

Figure D.33: *Additional sensitivity and bias plots for graph "West4" and batch execution "West4 Large".*

Glossary

agent function is the set of rules (or algorithm) that determines what move or rotation an agent will take given a trial number; the right turn, follow, phototaxis, and lateral phototaxis probabilities; and sensor readings. The trial number seeds the random number generator that the agent function uses to make comparisons against the probabilities to determine whether to make a clock-wise rotation or scan, follow corners, or turn toward light. ix, 29, 30, 32–34, 326, 329

batch execution is `num_trials` world executions parameterized by parameter values. All world executions in a batch share the same parameter values but each is given a unique trial number in $[0, num\_trials)$. xxiv, 4, 5, 19, 23, 29–31, 36–40, 58, 76, 77, 323, 324, 326

chart is a box and whisker plot showing the distribution of steps in which robots left certain regions. The regions are shown in columns. ix, 45, 47

data set is a set of paths that record the locations of robots during the execution of one or more worlds in a batch execution. 19, 32, 35, 37–39

discriminatory value is a metric that measures how well a prediction occupancy matrix discriminates between occupied and unoccupied regions in a target occupancy matrix. 64

excluded and overestimate heat map is a heat map that shows the regional excluded and overestimate values for the target occupancy matrix and prediction occupancy

matrix of a step for a batch execution. 61

follow probability is the probability of turning to the right or left if an agent is obstructed ahead but only to one side. 324

fp is an alias for the follow probability. 31

global position is the position of the region in which a robot is located. The global position is specified formally using coordinates of the region, relative to the origin at the south-west most region. Global position coordinates are given in roman numerals. 37, 325

graph refers to a connected, directed graph where each edge is associated with a single value. The edge values indicate the proportion of robots that will move between the source and destination vertices in one step. A graph represents the movement of proportions of robots within and out of a region. Graphs differ by their vertices and edges, providing diverse representations of the flow of robots through the regions of worlds in batch executions. vi, 3, 4, 23, 24, 35, 47, 48, 50–55, 76, 192, 328

heat map is the visualization used to show the distribution of robots over the regions of a target occupancy matrix or prediction occupancy matrix using monochrome shades. vi, ix, 40, 42, 46, 48, 60

histogram is a mathematical representation showing the frequency of steps for which robots left each region. vi, x, 39, 43–46, 60, 61

included heat map is a heat map that shows the regional included values for the target occupancy matrix and prediction occupancy matrix of a step for a batch execution. 62

inclusion is a metric that measures how well a prediction occupancy matrix contains the swarm in a corresponding target occupancy matrix. 64

lateral phototaxis is, in our worlds, motion toward a light source at the north end of the world when oriented east or west. 325

lateral phototaxis probability is the probability of lateral phototaxis. 325

local position is the position of the square in which a robot is located relative to the region in which the square is found. The local position is specified formally using the coordinates of the square, relative to the origin at the south-west most square of the region. The coordinates of a local position are given in arabic numerals. 37, 325

location specifies the precise state of a robot at a given step. A location is given by the global position, local position and orientation. 37, 323, 326

lpp is an alias for the lateral phototaxis probability. 31

macroscopic model is any of a number of different mathematical representations of swarm behavior from the swarm robotics literature where swarms are tracked collectively and the interactions between agents and between agents and the environment are not represented explicitly. 2, 3, 12, 15, 19

microscopic model is any of a number of different mathematical representations of swarm behavior from the swarm robotics literature where swarms are tracked as individuals along with the interactions between individuals and between individuals and the environment. 2, 3, 12–15

object is a target, obstacle, or robot that can occupy a square. 27, 326–328

obstacle is a non-taggable object represented in a square by the character 'O'. While no longer taggable, and no longer detectable by heat, they remain obstacles to robot movement. 27, 325, 327, 328

occupancy matrix is a one-dimensional matrix in which each cell represents the proportion of robots in a vertex of the graph of a region of the world. There is also one additional cell that is a sink where the proportion of robots that leave the world accumulate. vi, 3, 4, 58–60, 328

orientation specifies the direction that a robot is facing, which can be one of north ('N'), south ('S'), east ('E'), or west ('W'). 37, 325, 327

parameter value is an input for a batch execution such as the heat range, sonar range, maximum number of steps, or number of trials. The parameter values determine the number of trials for a batch execution, the maximum number of steps and sensor ranges for a world execution, and the probabilities used by the agent function. xxiv, 4, 5, 38, 76, 77, 192, 323

path is the record of the squares a robot occupied along with the corresponding orientations. Formally, a path is the step of the first location followed by the sequence of locations. 32, 35, 37, 38, 323, 326

path length is the length of a path or path prefix. It is the number of locations in a path or path prefix. 39, 45

phototaxis is motion toward a light source. In our worlds, it is motion toward a light source at the north end of the world when oriented south. 326

phototaxis probability is the probability of phototaxis. 326

pp is an alias for the phototaxis probability. 31

prediction occupancy matrix is a matrix that represents the predicted occupancy for a batch execution in a step, where for every region of the world there is a cell in the matrix that represents the proportion of the swarm predicted to occupy that region in that step. vi, 4, 23, 40, 58–60, 63, 323–325, 327, 328

prediction swarm is the swarm in a prediction occupancy matrix. The distribution of a prediction swarm in a prediction occupancy matrix can be compared to the distribution of a target swarm a target occupancy matrix using the predictive value metrics. 63, 328

region is a 5x5 square partition of the world. Every square, with the edge squares being the only exception, belong to a region. Regions do not overlap. 3, 23, 28, 54, 323, 324, 326–328

right turn probability is the probability of a clock-wise rotation or scan. 327

robot is an embodied, situated agent in a world. Robots move through worlds avoiding obstacles and tagging targets. Robots are represented in a square by an orientation that describes the direction in which the robot is facing. 3, 23, 27–30, 32, 35, 37, 323–326, 328, 329

rtp is an alias for the right turn probability. 30, 141

square is an atomic unit of a world that can be occupied by at most one object in any step. 26, 27, 325–328

square pattern is a 5x5 square grid, the squares of which may be unoccupied, targets or obstacles. A square pattern is a blueprint for the placement of targets and obstacles within the regions of a world. Square patterns differ by the obstacles and targets in their squares. 3, 4, 17, 20, 21, 27, 28, 192, 329

swarm is all of the robots in the world collectively. More generally, a swarm is a large multi-agent system that exhibits swarm intelligence. 1–3, 8, 325, 328

swarm intelligence is collective intelligence resulting from the interactions between agents and between agents and the environment. 1, 328

swarm robotics is the field of research and emerging engineering discipline that aims to study, design, and implement swarms. 1, 7, 325

target is an object that generates detectable heat and that can be tagged. Targets are represented in a square by the character 'T'. Robots search for targets. When tagged by robots, targets become obstacles. 3, 27, 325, 327–329

target occupancy matrix is a matrix that represents the actual occupancy in a batch execution in a step, where for every region of the world there is a cell in the matrix that represents the proportion of the swarm that is actually in that region in that step. vi, 24, 35, 37, 39–41, 60, 63, 323–325, 327, 328

target swarm is the swarm in a target occupancy matrix. The distribution of a target swarm in a target occupancy matrix can be compared to the distribution of a prediction swarm in a prediction occupancy matrix using the predictive value metrics. The target swarm does not include robots that are outside of the world. 63, 327

transition matrix is square matrix used to transform one occupancy matrix into another using matrix multiplication. Transition matrices are built automatically from graphs. A transition matrix has one row and one column for every vertex of the graph of each region. vi, 54, 55, 58, 59

world is a grid of squares. This grid is the environment where robots move, avoiding obstacles and tagging targets. Worlds are partitioned into uniform regions and targets

and obstacles are arranged in the regions according to a single designated square pattern. 3, 19, 23, 26–28, 32, 35, 76, 323–329

world execution is the evolution of a world from step to step for up to `max_steps` using the agent function. A world execution can end in fewer than `max_steps` steps if the robots tag all targets or all robots leave the world. Every world execution has a unique trial number that is used to seed the random number generator of the agent function. 3, 4, 19, 79, 323, 326