ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria ed Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

ENGINEERING BEHAVIOURAL DIFFERENTIATION IN ROBOTS CONTROLLED BY BOOLEAN NETWORKS

Tesi in
SISTEMI INTELLIGENTI ROBOTICI

Relatore
Prof. ANDREA ROLI

Co-relatore
Dott. MICHELE BRACCINI

Presentata da
ALESSANDRO CEVOLI

Anno Accademico 2018 – 2019

KEYWORDS

Automatic Design
Behaviour Differentiation
Boolean Networks
Robotic Agents
Stochastic Descent Search

Ai miei nonni materni, Ada Maria e Gino
To my grandparents, Ada Maria and Gino

Index

Introduction

The automatic design of robot control software is an important subject in current robotics; it aims to overcome the characteristics of classical human design, which might result limited and biased, in those cases where the robot task or the environment (in which the agent is situated) can not be formally specified or defined in advance or when the designer wants to explore new solutions. Indeed, automatically exploring the space of possible solutions and pin-point good ones through a series of constraints or performance requirements that the agent needs to satisfies, like an evolutionary algorithm may do, opens different possibilities to the design of the robotic agent controller, morphology and embodiment that we human would have never thought of. Moreover, such approaches are able to bring forth many necessary properties for agents, like greater degrees of autonomy, robustness and adaptiveness, in a more natural and far flexible way, whether they are on-line or off-line design procedures. Also, automatic design makes it possible to explore and exploit new kinds of control software, mainly those inspired by biological models.

For example, biological cellular systems offer an example of robust, adaptive and dynamic models where the tight link between artificial intelligence and dynamical system can be exploited. Boolean networks are a gene regulatory network model introduced by Kauffman [1] which are quite compelling from an engineering point-of-view, due to their rich and complex behavioural expressiveness despite the compactness of the model. Moreover, Boolean networks are able to reproduce many cellular systems properties and dynamics [2] like: different degrees of differentiation, stochastic and deterministic differentiation, limited reversibility, induced pluri-potency and induced change of cell type. Also, it has been shown that behaviours of Boolean network-based robots can be decomposed into smaller elementary behavioural blocks [3], each represented by attractors in the state-space of the dynamic system, connected by trajectories, and controlled by specific inputs.

The main goal of this thesis is to find an automatic procedure that is able to develop, given an initial Boolean network, a robot controller capable of differentiating between two or potentially more behaviours, after certain environmental signals are perceived by the agent sensors. Obviously, the more

variegated the external stimuli are, the more behaviours are more likely to be associated. Moreover, as hinted and exploited in some existing works [4, 5, 2], we assume that the environment is initially permeated by some kind of *noise* that does not allow the robotic agent to stabilize on a single behaviour but instead equally display all of them.

The major change from other works found in literature, like those by Roli et al. [6, 7], is that, once the agent has adapted for a certain environment, it should be able to go back to its previous pluri-potent state or manifest another behaviour as the environmental conditions change. That is, it should be able to re-differentiate itself as the environment changes.

In Chapter 1 we will initially provide the background and then discuss the state of the art from which this thesis stems.

In Chapter 2 we will talk about the strategies and methodologies adopted to tackle our problem of designing a robot controller based on Boolean networks and capable of re-differentiating. Then move on, describing how network performance is measured in order to validate the effectiveness of the devised design algorithms.

Finally, in Chapter 3 we will describe the various experiments and environment set-ups, the adopted agent embodiment and, finally, results on each network design, development and test.

Chapter 1

Background and State of the Art

In this chapter we will introduce the main works that have mainly contributed to this thesis, either from a theoretical and methodological or technological perspective. The aim is both to provide the background for the subsequent chapters, and to summarise the state of the art of the main topics covered in this thesis.

1.1 Gene Regulatory Networks and Boolean Networks

Gene regulatory networks, GRNs in short, are synthetic biology models that have been shown to reproduce most of the dynamics and interactions among genes (*Gene Expression*) in a cellular system. Among all the dynamics, they are able reproduce the Cell Differentiation and Morphogenesis processes:

1. Cell Differentiation is the biological process where different genes activation patterns enable cells to undergo differentiation from a pluripotent/toti-potent/multi-potent state to a more finalized and mature state by following a path along the linage tree.

2. Morphogenesis is the biological process where the different genes activation patterns enable an organism to develop from a more primitive shape to a more functional and adapted one. Instead of differentiating its functional properties and behaviours, the target of the process is the embodiment of the organism, its shape, sensory organs and "actuators".

Boolean networks are a gene regulatory network model introduced by Kauffman [1] which are quite compelling from an engineering point-of-view, due to

their rich and complex behavioural expressiveness despite the compactness of the model.

A Boolean network, BN in short, is a discrete-state and discrete-time dynamical system, represented as an un-weighted and oriented graph where each node is defined by a binary state $x_i \in \{0, 1\}$ and is associated to a Boolean function f_i. Each node represent the expression (1) or suppression (0) of the i-th gene. The state of the network at a certain instant t is defined as an ordered vector $S(t) = [x_1(t), ...x_n(t)]$, where $x_i(t)$ is the state of node i at instant t. The state of each node is updated deterministically or stochastically, synchronously or asynchronously, at each time-step by the output of the Boolean function $x_i = f_i(x_{ie_1}(t), ..., x_{ie_\mathtt{K}}(t))$, where $x_{ie_1}, ..., x_{ie_\mathtt{K}}$ are the states of neighbour nodes $e_1, ..., e_\mathtt{K}$ to node x_i. The parameter `K` defines the number of incoming edge to each node and can be different for each one of them.

A randomly generated Boolean network, also called RBN, is a BN where both topology and Boolean functions are randomly generated. Alongside the parameters $\mathtt{N} \in \mathbb{Z}$ and $\mathtt{K} \in \mathbb{Z}$, defining respectively the number of nodes and the number of neighbors to each node, the generation employs a parameter $\mathtt{P} \in [0.0, 1.0]$ that biases the output of each truth-table entry of the Boolean function f_i during its generation. The higher `P` is, the easier is to assign the output True to an entry of the truth-table.

The state of a cell can be represented as one of the attractors in the state-space of the dynamic system. A transition between cell states (n-potent to more specialized) corresponds therefore to a transition between two attractors.

An attractor is a cyclic sequence or pattern of states[1] in the network state-space that represents a stable state of the dynamic system (e.g.: a cell). Starting from any state of a BN, after a number of updates, hence a sequence *transient states*, the network will reach an attractor of some kind. It's called *trajectory* the sequence of transient states followed by attractor states. The set of states that leads to the same attractor is called *Basin of Attraction*. This means that all the possible gene expression patterns constitute the state-space of the BN and, among them, those in stable equilibrium are attractor states and their gene expression profile determine the observable cell type.

Normally a BN is an isolated system, which means that its state is not influenced by external factors: that is, once an attractor is reached, the trajectory will repeat the sequence of states in the attractor forever. But, by applying a certain level of noise that randomly flips the state of some nodes in the network, the trajectory may exit from the *attractor* and move from one stable state to another through a series of transient states in-between. This behaviour can be associated to tumor-like cells [8].

[1] 1 state = fixed point, n states = limit cycle, else strange attractor

1.2 Applying the Synthetic Biology and GRN Models to the Robotic Field

Cellular systems are robust, adaptive and dynamic, therefore the tight link between artificial intelligence and dynamical systems can be exploited. Moreover, it has been shown that behaviours of Boolean network-based robots can be decomposed into smaller elementary behavioural blocks [3], each represented by attractors in the state-space of the dynamic system, connected by trajectories, and controlled by specific inputs like chemical signals.

The survey [9] by Braccini reviews different methods in synthesizing robotic agents for different GRN-based models. The survey explores three main approaches: those which evolve or generate only the agent controller, those which evolve only the robot morphology, hence the embodiment, and those that attempt to achieve both by co-evolving agent embodiment and brain.

In the former, the prominent examples reported are:

- Eggenberger's Artificial Evolutionary System (AES) [10] which aims to control the main developmental process of the agent controller (in the article a neural controller) by exploiting biological processes like Cell Differentiation, Division and Adhesion, and concepts as Regulatory Units and Transcription Factors, Cell Adhesion Molecules and Cell Receptors. Everything in the model is encoded by an artificial genome composed of Regulatory Units and Control Genes, where the first activate or inhibit the seconds, while the latter modulate the developmental processes by producing substances regulating the activation of the aforementioned biological mechanisms.

- Another model, based on the biological principle of the *proteins synthesis regulation* [11, 12], develops an artificial neural network to control a robotic agent through a morphogenetical process that evolves the shape of the network. The employed evolutionary algorithm defines both topology, learning rules and weights of the networks, allowing it to synthesize any kind of network.

- The latest mentioned and explored approach is the one presented by Roli et al. [6]. In the article an effective automatic (meta[2]) procedure to design Boolean networks as robotic agent controllers is presented. The design process of the controller is modeled as a search problem, exploiting meta-heuristics, with the goal of minimizing the error in performing the

[2]Since it's not bounded nor restrained to the usage of specific meta-heuristic or search algorithms

given tasks of phototaxis and anti-phototaxis. The two tasks are alternated by a sharp sound signal that has to trigger the behaviour switch. The automation approach can be described as follow:

1. `generator` – The network is randomly generated, given the number of nodes N, the node arity K and a bias P which is used to randomly generate the output values of the entries in the truth-table of each Boolean function.
2. `evaluator` – The network is simulated and then evaluated on the basis of the chosen target requirements (objective function).
3. `meta-heuristic` – Given the results of the objective function, the meta-heuristic process starts the search. In the specific study the used search algorithm is a simple stochastic descent. The search process may lead to change the internal BN structure by modifying an entry of a truth table[3], if the evaluation output doesn't satisfy the minimization targeted value.

The article also propose an encoding for a BN-robot that tightly couple the BN and agent morphology/embodiment:

- The generated BN has synchronous and deterministic update, and is synchronized with the robot.
- For each BN a sub-set of `I` nodes is chosen as input nodes while another sub-set of `O` nodes as output nodes.
- The sensor readings are binary encoded and dynamically update the state of the input nodes.
- At each step the network is updated with the sensor readings, then it consequently updates its internal state and, finally, the state of the output nodes is applied as control value on the robot actuators.

In regard to the morphogenesis of a robot embodiment, the only example reported is always from Eggenberger [13]. In the article the previous AES model is extended by introducing positional information and pattern formation in the developmental process. This way each cells acquire a positional identity (coordinates) which changes the way the cell interprets the information, accordingly to its genetic constitution.

Finally, among the techniques belonging to the third approach, i.e. the development of both morphology and controller, the most relevant analyzed

[3]Potentially, the topology can be changed as well

is the Artificial Ontogeny [14] which combines ontogenetic development with genetic algorithms in order to evolve a complete agents. In such model, each genome is evolved using a genetic algorithm and treated as a GRNs, each gene produce "gene products" that either have a direct effect on the phenotype or regulate gene expression. Therefore the whole ontogenetic process enables a translation from a genotype (agent genome) to a phenotype (3-dimensional agent), later evaluated in a virtual environment. Each agent starts its ontogenetic development as a single structural unit then developed through a genetic algorithm tinkers on its genetic compound. On the morphological side, each unit has joints to attach itself to other units, carries a copy of the genome and has 6 diffusion sites, which contains any number of diffusing "gene products" and/or be connected to sensors, actuators or internal neurons. In addition to its embodiment, the genetic process develops the agent neural structure as well: of 24 different "gene products", 2 affects which growth units to diffuse into, 17 modulate the growth of the agent neural substrate and 5 genes are devoted to control gene expression.

The aforementioned article from Roli et al. [6] is further inquired in a study on the properties of artificially evolved Boolean networks [7]. In this article a greater focus is given to random Boolean networks in the *critical regime* with `K` $= 2$ and `P` $= 0.5$ [15, 16], those on the boundaries between order and chaos. This kind of networks display important properties such as *capability of balancing evolvability and robustness* and *maximizing the average mutual information among nodes*. The exploration of this boundary is a main difference from the previous work where the Boolean network controllers were initially generated with $K = 3$: while still in the *critical regime*, they are already more on the chaotic side.

The extracted evaluation features proposed in the article are:

- *State number* and *Frequency of state occurrence in sample trajectories* – The number of unique states in the collection of trajectories is an index of the state-space occupied by the dynamics. As such, the smaller number of unique states is, the greater the generalization capabilities of the network are, since it denotes trajectories that share a large number of transitions which means that *the system was able to generate a compact model of the world*.

- *Number of fixed point attractors* – Is a measure of the generalization capabilities of the system since fixed point attractors represent functional building blocks of the type `while <c> do <a>`. As such, emergence of such attractors means that the Boolean network was able to extract and exploit regularities inside the environment and classify them.

- An estimate of the *statistical complexity of the Boolean network*, hence to which extent the system is working in the critical boundary, is given by means of the LMC complexity. The complexity measure is calculated as $C(X) = H(X) \cdot D(X)$, where $H(X)$ is the *Shannon entropy*, which should be higher for complex networks since they display highly diversified trajectories, and $D(X)$ is the *disequilibrium*, which should decrease for complex networks since it estimates to which extent a systems exhibits patterns far from equi-distribution, thus presenting trajectories composed by repetition of few states.

Each feature is calculated by analyzing the BN-robot trajectories in the BN state-space and extracted for each evolutionary step.

The results brought by the article shows that robots, which successfully achieve the target behaviour, are characterized by a decreasing number of unique states as the complexity grows. Also, both *entropy* and *disequilibrium* show the expected complementary results for successful networks, leading to a steadily increment of the complexity during the training phase.

A similar and somewhat complementary automatic design approach to [6], concerning the development of Boolean networks with a given set of target properties, but regarding the field of synthetic biology, Cell Differentiation in the specific, is described by Braccini [16, 17]. There, two approaches, both based on stochastic descent, are defined:

- Adaptive Walk – A simple stochastic descent search algorithm where, once a BN is generated, it is iteratively modified by flipping one random entry of the truth-table of one random node in the network, until either the evaluation of the network through an objective function matches the target value or the algorithm reaches the maximum number of iterations. The algorithm allows for *sideway moves*, that is, flip moves that generate a network with the same objective score to the previous one. This allows the algorithm to explore eventual plateaus in the search space.

- Variable Neighbourhood Search – A more complex search algorithm where the number of flips increases overtime until a better solution than the current one is found. The stop criteria are the same of the Adaptive Walk. The compelling features of this algorithm is the previously described *sideway moves* plus the ability to escape from local minima which, in a random search, are very likely to happen.

In the article [3] is inquired the aforementioned direct mapping between attractors and robot behaviours. The main goal of the paper is to exploit the

properties of the attractor landscape in order to control the speed of a robotic agent while performing a simple behaviour like phototaxis. The mapping between the elementary blocks composing the behaviour and the attractors is achieved by means of an adaptive process (a simple stochastic descent). Moreover, in such setup, some nodes of the network are directly coupled with the embodiment: some nodes are temporarily switched on by what is perceived by the sensors, while other nodes are chosen to directly control the actuators output.

The BN of the first experiment is characterized by 2 control genes: one assessing the presence of light, while the other is switched on in order to suppress the behaviour as the agent comes too close to the light source, hence stopping the agent in its proximity.

In the same paper is presented another experiment where some nodes, chosen as receptors, are permanently clamped to 1 or 0 as a consequence of an external signal. The clamping leads to changes in the transition graph of the BN, cutting off some transitions and therefore the expression of some attractors. Moreover, the action of clamping opens the networks to the expression of *conditional attractors*: (possibly new) attractors that are conditioned by an external factor (clamping), which impose the agent to specific attractors composed by states with a common pattern.

1.3 Responses of Boolean Networks to Noise

Fretter [5] has explored the probability ρ that a Boolean network (immersed in a noisy environment), once reached an attractor, returns on the same stable state after that h nodes have been perturbed. The experiment hence aims to provide some tools to describe quantitatively the robustness of the network. In the paper, various type of networks are studied: independent nodes, simple loops, collection of single loops and RBNs. For RBNs, different `K` are used: $K = 1$ for frozen phase networks, $K = 2$ for critical networks and $K = 3$ for chaotic networks. In this cases, the results show that for $K = 1$ the probability ρ varies widely, and is easily larger than $\rho = \frac{N}{2}$. On the other hand, chaotic and critical networks show interesting results: *RBNs with $K = 2$ display lower robustness than those with $K = 3$*; in the first case ρ can be described as $\rho = (aN^{-\frac{1}{3}})^h$, while for chaotic network ρ rapidly decrease for small h and then sits on a plateau always around $\rho = \frac{2}{3}$.

This difference between critical and chaotic RBNs can be attributed to the difference in the number of attractors between critical (higher) and chaotic (lower). It is therefore easier for RBNs with $K = 3$ to go back to the same attractor once perturbed since more states will belong to the same basin of

attraction.

In the end, all the result have been achieved by means of randomly generated networks but, generally, the topology of the network has a consistent influence on the on the dynamics of the network and, therefore, on the number of expressed attractors.

For low level of noise (perturbation ~ 1 node at time), it is possible to draw an *attractor graph* and the correlated *Attractor Transition Matrix*, where for each pair of attractor (a_i, a_j) is assigned a probability ρ describing how easily, in presence of noise, the network can move from one stable state to another. An effective metaphor describing the *attraction basins*, called the "Epigenic Landscape", has been provided by Waddington [18]: the network state is like a marble that rolls down the landscape topology until it reaches a local minima, that is, an attractor.

A mathematical model, called the Threshold Ergodic Set (TES_θ), reproducing the abstract properties of Cell Differentiation[4] has been proposed by Serra and Villani [2]. In this model the notion of Ergodic Set previously proposed by Ribeiro and Kauffman [4] is extended by means of a threshold $\theta \in [0, 1]$ that cuts off from the *attractor graph* those transitions (between attractors) such that $\rho < \theta$. An Ergodic Set is a subset of the all possible attractors that entraps the system for $t \to \infty$. Since there is usually only one of such sets for most Noisy (Random) Boolean network, the proposed model introduce the threshold θ in order to split the *attractor graph* into smaller (possibly isolated) TESs, till a single TES is composed by a single attractor. Therefore, a TES_θ is a ES where attractors are directly or indirectly θ reachable.

The similarity with the biological model and TES_θ is that wandering through a large number of attractors could be associated to a toti-potent cell while, as the threshold increases, smaller TES_θ appears which corresponds to more differentiated biological forms. The TES composed by single attractors are, therefore, the fully differentiated cells.

Since different noise levels are usually associated to different level of differentiation, the increasing of the threshold θ would correspond to a decreasing level of noise in the biological environment.

[4]Different degrees of differentiation, stochastic and deterministic differentiation, limited reversibility, induced pluri-potency and induced change of cell type

Conclusions In the end, many have dealt with the problem of automatically design robot controller based on synthetic biology models; among them, Boolean networks are interesting due to their simple model, robustness, adaptiveness and dynamism typical of cellular system, and the ability to reproduce some of the cell properties and dynamics. The usual approach to the design process is to employ meta-heuristic strategies based on stochastic descent and therefore to randomly explore the solution space. The results of the design processes appear promising especially when evolving the sole Boolean function paired to each network node, while keeping intact the network topology. Furthermore, if noise is exploited in the design process, it may open new possibilities to generate networks with certain level of robustness.

In the next chapter we firstly connect together many of the methodologies and information expressed in this Chapter, through the analysis of these various articles, then improve with novel approaches those which will be employed to achieve the primary goal of this work.

Chapter 2

Methodologies

In this chapter we introduce and describe the methods applied and solutions adopted to attain behavioural diversification in BN-robots. We identify as BN-robot any kind of robotic agent equipped with a controller based on one or more Boolean networks, where some node are employed as sensor receptors while others interface with the agent actuators.

The work is divided in two main parts: one involving the automatic design of a Boolean network able to achieve a specific non-trivial task and that could employed as a stand-alone robotic agent controller, similarly to Roli et al. [6]; while, in the other, we move to design a Boolean network that is able to express controlled re-differentiation by satisfying a certain number of constraints. Such BNs will be employed to compose a hierarchical control unit that differentiate into any given number of behaviours.

A bit of terminology Before going on, it would be useful to specify and clarify some of the terminology that will adopted from now on in this work.

The words agent, robotic agent and robot are used interchangeably. It is always implied that the agent is a robot equipped with some kind of Boolean network-based controller.

As previously stated in [3], a direct mapping between network attractors and robot behaviours can be achieved; as such, the words *attractor* and *robot behaviour* will be used interchangeably.

When talking about the *display or expression of a certain attractor* on a given BN, we imply that the network state is updated for some time until an attractor is displayed.

For *noise phase* and *transition phase* we mean those time-steps where the BN is affected by perturbations due to the presence of *noise* or when, due to some changes in the environment, the network move from one attractor to another. These *transitions* between attractors, called *transients*, are usually

compound of states that are not part of an attractor *per se* but uniquely lead to it (when not under noise).

While discussing the simulation of the network, either to evaluate or test it, we refers to the triplet *agent initial position*, *agent initial rotation* and *<type> source initial position* as *starting configuration*.

2.1 The Adopted BN Model

All Boolean networks in this work are random Boolean networks – `RBNs` [19, 17] characterized by *synchronous and deterministic update sequences*, like most of the networks explored in the articles presented in Chapter 1, and described by the following main parameters:

- `N` $\in \mathbb{Z}$ – the number of nodes in the network.
- `K` $\in \mathbb{Z}$ – the number of neighbours/predecessors of each node. Hence the number of inputs to each associated Boolean function.
- `P` $\in [0, 1]$ – the bias to assign either `True` or `False` to an entry of any Boolean function truth-table in the generated `RBN`.
- `Q` $\in [0, 1]$ – the bias to assign either `True` or `False` to a node as initial state when the `RBN` is generated.

While not explicitly defined in their signature, the model of all the networks also support noisiness – `NRBNs` [19, 17]. The level of *noise* in the network is regulated by means of a parameter $\rho_{noise} \in [0, 1]$.

An `Open Boolean Network`, `OBN` in short, is a kind of BN which have some nodes assigned as receptors and others as actuators [6]. They are characterized by two more control parameters, named `I` $\in \mathbb{Z}$ and `O` $\in \mathbb{Z}$, which define respectively the number of input and output nodes in the network. We identify as input nodes those nodes which will be employed to receive feedback from the environment, while output nodes are those adopted to communicate or interact with the environment. This is the model on which any Boolean network controller in this work is based upon, therefore all the developed robotic agents that will be developed are indeed BN-Robots.

For simplicity, input and output nodes are chosen sequentially inside each network; that is, if the network has $N = 20$, $I = 8$ and $O = 2$ then $nodes \in [0, I)$ are input nodes, while $nodes \in [I, I + O)$ are output nodes.

2.2 Goals and the Biological Idea

The main goal of this thesis is to find an automatic evolutionary or search procedure that is able to develop, given an initial Boolean network, a robot controller capable of differentiating between two or potentially more behaviours, when certain environmental signals are perceived by the agent sensors. Obviously, the more variegated the external stimuli are, the more behaviours are more likely to be associated. Moreover, as hinted and exploited in previous works [4, 5, 2], we assume that the environment is initially permeated by some kind of *noise* that does not allow the robot to stabilize on a single behaviour but instead (more or less) equally display all of them.

On a biological level we could describe the robot dynamics as a pluri-potent stem cell that is initially immersed in a solution (environment) characterized by highly mutability and dynamism, such that the cell is not able to find a stable pattern of signals on which develop itself among the chaotic series of stimuli it receives. Then, as the *noise* subdues and the environment becomes more and more ordered, the cell is able to determine a pattern in the series of signals[1] and develop itself, evolving along a certain branch of its lineage tree.

Until this point the work described mostly follows the tracks already laid down by Roli et al. [6, 7] and correlated works. **The breaking change from the previous material that we aim to achieve is: once the agent has adapted for a certain environment, it should be able to go back to its previous pluri-potent state or manifest another behaviour as the environmental conditions change. That is, it should be able to re-differentiate itself as the environment changes**.

This dynamics can be associated (to a certain degree) to the behaviour of a cancer cell [8] which, if stressed with enough stimuli, could undergo a re-differentiation process that brings it to a cancer state, which may be seen as a pluri-potent stem-like state. What we want more is for it to even develop to a completely new non-cancer cell.

In order to achieve what expressed above, the work has been mainly cut down into three sub-problem:

1. First, devise an automatic design procedure that generates a single `Open Boolean Network` satisfying the following constraints:

   - The network has only `M` attractors.
   - While perturbed, for each attractors pair (a_i, a_j), the transition probability $\rho_{ij} \mid a_i \rightarrow a_j$ must be greater than a given threshold τ_{ij}.

[1] Like receiving feedback all from the same kind of source, will it be heat or electrical

- While under the effect of *noise* the network displays all the attractors, independently from their frequency.
- While unaffected by noise, given a specific input stimuli, the network displays always the same attractor regardless of the initial state.

That is, they are able to express re-differentiation: to come back to their pluri-potent state or move to other attractors once the environmental signal change, while expressing all their possible behaviour and displaying a selected number of behaviours, much like a cancer cell would do. This kind of networks usually have no output nodes and a number of input nodes `I` such that the possible stimuli to be perceived are $2^{\mathtt{I}}$.

Such Boolean networks, will be either referred to as `Control-BN` – `CBN` or `Selective-BN` – `SBN`, interchangeably, in order to distinguish them from other Boolean networks.

2. Then, with another ad-hoc automatic procedure, generate an `OBN` achieving a very specific behaviour. Such kind of networks will be called `Behavioural-BN` – `BBN` in order to tell them apart from the networks specified on point `1`. They can effectively be adopted as stand-alone robot controllers.

3. Finally, achieve an efficient and flexible mapping methodology between the attractors expressed from the network developed in point `1` and the behaviors expressed by the networks evolved in point `2`, that is: mapping an attractor to another whole Boolean network.

We have chosen to do so in order to, from an utilitarian point-of-view, simplify both evolutionary procedures while, on an engineering point-of-view, achieving flexibility and generality of the generated networks: instead of developing a whole network which attractors represents the wanted ad-hoc behaviours and satisfying the cancer-like constraints, it is far more valuable, especially for testing and study purpose, to have a generic network which presents the said cancer behaviour while its attractors can be whatever and only ontologically associate them with a given behaviour.

The whole structure, composed by a single `SBN` and `M` different `BBNs`, will be referred to as `Selective-BN-Controller` – `SBNC`.

2.3 From Attractors to Behaviours

Starting off from the last on the three sub-problems, we will point out the main idea behind the chosen mapping method.

Given a produced `Selective-BN`, such network will display the given number of `M` attractors, each of variable length, and its *Attractor Transition Matrix* will display, for each element, a transition probability ρ_{ij} greater than the specified threshold τ_{ij}. Being `M` the number of possible assignable behaviours, then each state composing one of the `M` attractors should uniquely lead to a specific behaviour. But, in doing so, which behaviour to choose when the network is not on an attractor? More specifically, since the network has to equally display each behaviour when affected from noise, how to map each behaviour into a network state when this is not part of an attractor?

There are two possible solutions to such dilemma:

- *State graph* approach – One is to simply assign a behaviour to a known attractor and, based on the complete state graph of the developed network, unambiguously map each state in the graph to the behaviour associated to the attractor the graph refers to. While doable, the approach seemed quite expensive computationally and memory wise since the graph size may not be trivial and its complexity is $\mathcal{O}(2^N)$.

- `ATM` approach – As before, simply map each behaviour to a known attractor but, this time, each state composing the attractor uniquely leads to the mapped behaviour while, for non-attractor states, randomly choose which behaviour to express by weighting each choice with the transition probability $\rho_{ij} \mid a_i \rightarrow a_j$ described in the network *Attractor Transition Matrix*. Since non-attractor states will appear only during *noise phase* or in transients of a trajectory leading to an attractor, this approach works during both *noise phases* and *transition phases*, when the network is either regressing to a pluri-potent state or moving from one behaviour to another.

In the end, among the two above described methodologies, the chosen one is the `ATM` approach since it gives a more concise and simple solution, even if determining a network `ATM` may be an expensive operation for big Boolean networks. Also, while both approaches are equivalent when the network is stable on an attractor, the precise mapping between transients states and behaviours given by the *state graph* approach isn't needed. Moreover, realistically speaking, having transition phases is far more interesting than selectively changing attractors like an `if-statement` would, since this transition can be interpreted

as the adaptation process of the network to the new environmental conditions. Also, it better outlines the real behaviour of the network if the expressed attractors were (sort-of) the behaviours themselves.

Furthermore, in order for the mapping to be more flexible, instead of directly associating attractors to behaviours, *each behaviour is uniquely and a priori associated to a possible input stimuli* while, during the development phase of the `SBN`, each attractor is dynamically associated to the input stimuli that induce the network to transit always on that attractor, whichever the initial state of the network may be.

Still, letting behaviours being imposed by transition biases means that there is no certainty on which behaviour will prevail, on the short term, due to the randomness (especially in a noisy environment). Surely enough, in a long term run, the ratio of alternating between behaviours will definitely match that expressed by the `ATM`. Moreover, a factor to be wary about during behaviour analysis is that, since behaviours can vary greatly in what they do, some of them will surely negatively influence those who comes after. This is true for both aforementioned approaches. To give a proper example: given an agent that has to express both `phototaxis` and `anti-phototaxis`, it is possible that either of the two behaviours has more influence on the global conduct than the other, be it due to random fluctuations in the behaviour choice or an higher average speed of one of the behaviours[2]. As such, if `anti-phototaxis` is expressed first then it leads the agent further away from the light source which will influence negatively the evaluation of `phototaxis`.

[2]Since they are designed by an automatic procedure is not a possibility that can be discarded.

2.4 Automatic Design of a Selective-BN: Achieving Controlled re-Differentiation

In this section we'll move on, describing the approach undertook to tackle the problem of automating the development of the `Selective-BN`.

A `Selective-BN` usually has no output nodes but only a number of input nodes that depends on the environmental stimuli that guide its differentiation process. In our scenario the environmental signal is binary, hence a single input node with each stimuli leading to a unique behaviour: $0 \rightarrow$ `phototaxis` or $1 \rightarrow$ `anti-phototaxis`.

2.4.1 Constraints

As previously mentioned, the target Boolean network must satisfy four constraints:

1. The network expresses only `M` attractors, of any length, when situated in a noisy environment.

2. For each attractor pair (a_i, a_j), the transition probability ρ_{ij} for moving from $a_i \rightarrow a_j$, while the network is perturbed, must be greater than a given threshold τ_{ij}.

3. While under the effect of *noise*, the network displays all the attractors, independently from their frequency.

4. While unaffected by noise, given a specific input stimuli s, the network expresses always the same attractor a_i regardless of the initial state, after ϕ update steps.

   Note that, even if the number of stimuli is $2^{\mathtt{I}}$, it doesn't mean that the number of behaviours to be displayed must amount to that: `k` different stimuli may lead to the same attraction states, what matters is that all the attractors (`M`) are mapped into at least one stimuli.

   What we want is for the network to be robust enough to always return to the same attractor [5] but only for a certain set of stimuli localized on a defined portion of it topology (the input nodes).

   Also, the mapped stimuli will define to which behaviour the attractors are associated since the association stimuli-behaviour is defined a priori.

The constraints are each characterized by some control parameters:

1. `M`, the number of attractors the network should display under noise.
2. $\tau_{ij}, \ \forall \ i, j \in A$ [3], specifies a lower-bound for transition probability ρ_{ij} that must be met for each entry of the network `ATM`.
3. (a) `i`, the number of iteration for which the BN should be updated in order for all the attractors to surely appear at least once.
   (b) ρ_{noise}, the probability to apply *once* for iteration a perturbation on a random node. The perturbation consists in a flip of the state value in the chosen node, as similarly done in previous works [6, 5, 19]. The choice of the *noise* bias ρ is some what critical: high values of *noise* increase the difficulty in satisfying the `constraint #3`, which depends also on the length of the attractors of the network. If the length is short, like between $[1, 3]$ states, then it is easier to display them for higher level of *noise* since it's less probable for short sequence of states to be broken. Indeed, if *noise* is sure to take effect ($\rho = 1.0$), then it will be impossible to display any attractor with length greater than `1` and, even for such length, everything depends on the noise, which has to place the network in a state inside the basin of attraction of that attractor. Lower value of the bias make it possible to display longer attractor but, on the other hand, it make easier for the network to get trapped into an attractor for long times, letting it unable to express other attractors.
4. ϕ expresses for how many steps the input stimuli should be imposed on the input nodes of the Boolean network. This also sets a higher-bound to the number of steps the network has in order to adapt to new environmental conditions. Biologically this can be associated to a deafening of the cell to the stimuli.

Notably, the last of the constraints is the hardest to achieve since from [5] we know that networks with higher `K` tend to be more robust to *noise* with a probability to return on the original attractor (once perturbed) of approximately $\frac{2}{3}$. This means that for more chaotic networks is far more difficult to achieve `constraint #4` than for those in the critical boundary.

Moreover, the evaluation of `constraint #4` is quite expensive since it requires to test each possible starting state (2^N) to be properly achieved. In this work the problem has been simply brute-forced with some multi-processing since we where working on small enough networks.

[3] Where A is the set of attractors $\in BN$

While `constraint #2` is computationally very easy to achieve *per se*, it becomes computationally harder as $\tau_{ij} \to \frac{1}{|A|}$: more and more networks must be generated before one met this requirement since each $\rho_{ij} \to \frac{1}{|A|}$ for increasing τ_{ij}.

2.4.2 Network Design

Since the problem requires the satisfaction of a number of constraints, it can be modeled as a rather simple *Constraint Satisfaction Problem* – CSP. The employed solution to tackle with this kind of problem is a `Generate-And-Test` algorithm, given that we will explore small networks with $N \leq 10$, where the `generation` phase always produce a new `Open Boolean Network` at each unsuccessful iteration while the `test` phase verifies that all the above specified constraints are satisfied.

In our case study, the randomly generated `OBN` has the following properties: *each of its input nodes has at least one outgoing edge* and *any node is devoid of self-loops*.

Algorithm

A pseudo-Python code regarding the meta `Generate-And-Test` algorithm and the constraints is provided below.

```
def GenNTest(max_iters, generator, tester, evaluator):
  it, score, sol = 0, None, None

  while it < max_iters and not score:
    sol = generator()
    score = evaluator(tester(sol))
    it += 1

  return sol if score else None
```

```
def test_attractors_number(bn, M): # Constraint n.1
  return len(bn.atm) == M
```

```
def test_attractors_transitions(bn, taus): # Constraint n.2
  return all(bn.atm[i][j] > t for t in taus)
```

```
def test_space_homogeneity(bn, i, rho): # Constraint n.3
  states = [bn.noisy_update(rho) for _ in range(i)]
  return len(search_attractors(bn.atm, states)) == len(bn.atm)
```

```
def test_attraction_basins(bn, phi): # Constraint n.4
  attrs, params = dict(), product(states, inputs) # 2^N * 2^I

  for s, i in params:
    bs, as = test_state_attraction(bn, s, i, phi)
    if len(as) > 1 or bs in attrs and attrs[bs] != as[0]:
      return False
    attrs[bs] = as[0]

  return attrs if len(set(attrs.values())) == len(bn.atm) else False
```

Description

In `test_attractors_transitions`, the parameter `i` employed in our case study is dynamically set to:

$$i = 4 \cdot \max_{k=0}^{|A|}(|a_k|) \cdot N^2$$

where `N` is the number of nodes in the network and `max` returns the cardinality of the longest attractor of the network. The multiplicative factor 4 is used to slightly increase the number of iterations.

The `test_state_attraction` function, once assigned the new state to the network and applied the related input values on each the input node, updates the networks and scans its trajectory until an attractor is found. It then returns the given input and a list of found attractors. The list of attractors should contains only one element since we look for a Boolean network that for a given input, whatever the internal state may be, always moves into the same attraction sequence after at least ϕ steps.

The `if-statement` condition on `constraint #4` checks whether, for the same input, different attractors have been found or not; that is, if the network moves to different attractors from two different starting states for equal input values.

2.5 Automatic Design of a Behavioural-BN

At last, we will describe the chosen solution to resolve the automatic design of the behaviour-achieving network, which completes the robot controller as a whole. The design process is mainly composed of two phases: (1) a *development phase*, where the network is generated and improved through a `meta-heuristic` algorithm, and (2) a *test phase*, where the developed networks are further studied to determine their performances.

In the next subsections we will firstly describes the proposed `parametric Variable Neighbourhood Search` algorithm, `pVNS` in short, then move on, outlining how and which data are gathered in order to describe the proposed objective function and, finally, the employed evaluation methodology for both development and testing.

2.5.1 Network Development

In previously described works [6, 16], the problem of automatically design a Boolean network with certain given characteristics is modeled as a simple search problem and resolved by exploiting a `meta-heuristic` search algorithm. Both works indeed outline an approach quite similar to resolve the design problem:

- In [16], as mentioned in Chapter 1, is proposed an interesting `meta-heuristic` algorithm: a version of `Variable Neighbourhood Search` – `VNS`, hence based on stochastic descent, that incrementally improves a given solution until a target condition is met.
- In [6] an `evaluator` procedure is adopted to produce an objective function score based on a set of target requirements; such value is in turn given as input to the `meta-heuristic` algorithm that consequently modifies the network through a search procedure (for example a stochastic search like above). While the given description is rather generic, it matches both the previously described `Generate-and-Test` and the above `VNS` algorithms: (A) both adopt an `evaluator` function, the former to test the target constraints are satisfied while the latter to assign a score to a new solution. (B) Both try to improve the solution through some kind `meta-heuristic` strategy: the former generates a completely new solution on each iteration while the latter incrementally modifies a given initial solution. In the end, both methods adopt a stochastic descent search but applied on different levels of abstraction.

In this work the proposed approach arises by improving the `VNS` algorithm in order to adapt the solution to effectively tackle our problem.

The `Generate-and-Test` algorithm, adopted in Section 2.4, hasn't been considered since the problem at hand doesn't involve the satisfaction of some constraints, hence can't be modeled as a CSP, but rather concerns the evolution of an already existing solution to meet a certain level of performance in achieving a task. Moreover, even if this problem could be modeled as a CSP and resolved through a `Generate-and-Test` algorithm, it would be rather inefficient since on each iteration a new random network would be generated, losing all the good properties of a possibly good-but-not-perfect previous solution. On the other hand, the contrary could be a rather good alternative: tackling the CSP problem described in Section 2.4 with an `VNS` algorithm would probably result more efficient for larger networks since the solution would be improved iteratively rather than be quickly discarded.

Since the evolutionary procedure is done offline, each evaluation of the network will require a simulation on a robotic agent. As such, on each simulation some information will be gathered in order to evaluate the behaviour of the agent and produce the objective function score.

Algorithm

```
def ParametricVariableNeighbourhoodSearch(
    sol, target, evaluator, comparator, scrambler, tidier,
    max_iters, min_flips, max_flips, max_stalls, max_stagnation):

  score = evaluator(sol)
  it, flips, n_flips, n_stalls, stagnation = # Init variables

  while all(it < max_iters, not stop, not comparator(score, target)):
    sol, flips = scrambler(sol, n_flips, flips)
    new_score = evaluator(sol)
    if comparator(new_score, score):
      score = new_score
      stagnation, n_stalls, n_flips = # Resets values
    else:
      sol = tidier(sol, flips)
      stagnation, n_stalls = # Increase both by 1
      if n_stalls >= max_stalls:
        n_stalls, n_flips = # Reset stalls and increase flips by 1
      stop = # Checks if max stagnation or flips are exceeded
    it += 1

  return sol
```

Description

Highlighting the differences between the proposed `parametric-Variable Neighbourhood Search` algorithm and the original from Braccini [16] we can see that:

- The output of the evaluation function, previously defined as `distance`, is here renamed `score`. Its type depends on the implementation of the given `evaluator` procedure.
- The counter `noImprovements`, that would register the number of iteration without improvements before increasing the number of flips, is here renamed `n_stalls`.
- The parameter `max_flips` has been added in order to support different level of scrambling, from more coarse to finer grain.
- Another counter has been added, `stagnation`, that stops the algorithm when there have been a number of iterations without any improvement equal or greater to `max_stagnation`, regardless of `max_flips`. This also introduce a panic button that would stops development of networks that took too long for too little.

Furthermore, the proposed solution is indeed parametric:

- `evaluator` – Implements the strategy that tests and assigns a score to a solution. It takes the place of the `ComputeDistance` function and takes only the solution as parameter. This has been done to maintain the algorithm "compact" and more generic, but eventually more parameters can be given from outside by exploiting high-order functions.
- `comparator` – Implements the strategy that compares the found score with the given target value. Takes the place of the comparison between found distance and target distance, allowing more complex comparisons to be made. Depending on its implementation, *sideway moves* may be either allowed or not.
- `scrambler` – Together with `comparator` it represents and implements the `meta-heuristic` algorithm that adapt the network based on the score returned by the `evaluator`. It takes place of both the operations of `GenerateFlips` and `ModifyNetwork`, that is, it should do both: scramble the network until its changed. Like in [16], the function signature takes as input the previously applied flips in order to avoid repeating them between successive iterations.

- `tidier` – Takes place of `ModifyNetwork` when is used to restore the solution to a previous version. That is, if the `comparator` procedure has returned a score worse than the current one.

Before defining how the four parametric functions have been effectively realized, it is better to first define which data are collected from each simulation of the `Behavioural-BN`.

2.5.2 Design Methodology

Initial Solution

In our case study, similarly to networks developed by the previous automatic procedure, the initial solution is a random `Open Boolean Network` such that:

- Each input node has at least one outgoing edge and **none incoming**.
- Each output node hasn't edges outgoing to or incoming from other output nodes.
- No self-loops are allowed.

Target Behaviours

As already said, the target behaviours that the agents have to achieve are `phototaxis` or `anti-phototaxis`. We'll now describe what we expect from an agent achieving these behaviours:

1. An agent achieving `phototaxis` should move from any point of the environment, be it near or far away, to the closest possible proximity to the light source.
2. An agent achieving `anti-phototaxis` should, instead, move from any point of the environment, be it near or far away, to the furthest possible distance from the light source.

Robot Simulation

As previously mentioned, the evaluation of the network requires its simulation on a robotic agent for each step of the algorithm. Therefore, the objective function will base its evaluation on the data collected during such simulation.

The collected data obviously varies depending on the task to be achieved by the agent and its embodiment; getting this work as example, since the agents

have to learn `phototaxis` or `anti-phototaxis`, the data from each equipped light sensor are obviously required.

Some useful data to be collected are therefore:

- `<type>` *sensors data* – In our scenario, *light sensors data*.
- *source position* – useful to keep track of its position, especially if it's in motion but that is not our case.
- *agent position* – useful for multiple reasons, like plotting agent path in the environment or, if paired with the *source position*, to be used as corroborating factor for the evaluation score during either training or testing.
- *network state* – useful to study dynamical properties of the network and its attractors as in [7].
- *step number* – necessary to maintain order among the collected data and, knowing the simulation basic time-step, tell at which time of the simulation the data have been gathered.

Not all the data here described are necessary for the evaluation of the network behaviour but are indeed orthogonally useful for testing or later investigations. In our case study the only data used during network automatic development are the values perceived from the light sensors since we wanted for the agent to improve based on its own perceptions (*situatedness*).

The objective function Since in our work the agents have to achieve `phototaxis` or `anti-phototaxis`, there is need of a feedback that lets agent understand whether or not is doing its job right. The proposed objective function assigns a score s to *a single simulation run* by aggregating all the perceived *irradiance* measurement (in that run) as follow:

$$s = \left[\sum_{i=0}^{\max(step_s)} \max_{j=0}^{|S^{IR}|}(I_{ij}) \right]^{-1}$$

where $|S^{IR}|$ is the number of light sensors, while $\mathtt{I}_{ij}$ is the *irradiance* value perceived at step `i` by sensors `j`. Everything is elevated to `-1` in order to align *irradiance* and `distance` from the source, since $irradiance \propto \frac{1}{d^2}$; this way the score will diminish as the agent gets nearer to the light.

Note that optimal scores greatly depend on (A) how much *time* is given to the agent to perform its task and (B) the *intensity* of the light source[4]: more *time* means possibly higher scores with equal growth trends, but greater *intensity* leads to a faster increase of the score in equal time lapses, especially for agent far away from the light. Moreover, remember that for `anti-phototaxis` its score will continue to decrease as long as it's exposed to the light, since the max of the light sensor readings will be non-zero. The pace at which their score decreases depends on the distance at which the agent is and how fast it gets away. Therefore, on `anti-phototaxis` will be rewarded those networks that get far away from the source as fast as possible, which will reflect in an higher score.

The proposed score model must be indeed minimized for `phototaxis` while maximized for `anti-phototaxis`.

Another possible approach would have been to also sum all the perceived sensors values on one step:

$$s = \Big[\sum_{i=0}^{\max(step_s)} \sum_{j=0}^{|S^{IR}|} I_{ij} \Big]^{-1}$$

Such approach has been discarded since the local contribution from the single sensor would get lost. For example, given two agent learning `phototaxis`: one near the light but with few sensor oriented in its direction while the other more distant but with more sensors oriented to the source. There may be a situation where each single sensors of the nearer agent has single measurement values greater than those on the further agent but the global contribution of those on the latter are greater than those on the former. This would lead to reward more those agents that get more sensors exposed to the light source by maximizing their global contribution instead of simply going in the direction where the stronger perceptions are sensed.

Moreover, given an environment characterized by `n` light sources, with the proposed score model the agent is more likely to go towards the source with the stronger intensity or that seems to irradiate greater energy, while the other method would probably lead the agent to go in the (not precisely) middle of the `n` sources.

One more alternative approach would have been using the error function provided by [6] and using only its `phototaxis` or `anti-phototaxis` contribution:

[4] A multiplicative factor applied to the perceived *irradiance*.

$$E = \alpha(1 - \frac{\sum_{i=1}^{t_c} s_i}{t_c}) + (1 - \alpha)(1 - \frac{\sum_{i=t_c}^{T} s_i}{T - t_c})$$

This model hasn't been chosen in order to provide a different evaluation that would use only data local to the agent, that is, given by its own point-of-view on the world.

Scrambling, Tidying, Evaluation and Comparison

In our scenario the four parametric functions, which outline the automatic development algorithm overall behaviour, have been designed as follow:

- `scrambler` – This function, as previously stated, encapsulate both flips generation and network modification. Both generation and modification are rather trivial since they behave the same as in the work by Braccini [17]: the former produce a number of flips equal to the `n_flips` variable, while the latter apply these flips on the network.

  A flip is a pair composed of a node label and a truth-table entry and is applied on the network by negating the output value of the specified table entry. Moreover, in order to improve the stochastic search strategy, the input and terminal nodes[5] of the network are excluded from any flipping.

  In this case study, like in previous works [6, 16], *we will not explore the development of the network topology* but only of its internal structure.

- `tidier` – Like above, this function behave the same way as in the work by Braccini [17]: given a set of last flipped table entries, the function applies the flip on each specified entry. Since each entry is a binary value, the procedure returns the network to its previous incarnation by negating the actual entry value.

- `evaluator` – Like already said, an evaluation of the `Behavioural-BN` requires its simulation. The problem is that a single simulation is only able to test a single possible *starting configuration* of the agent[6]. That is, one simulation is insufficient to effectively check whether or not the network is able to successfully complete the target task. To this extent the employed approach has been, for each evaluation, to execute `T` simulations, each associated to a different *starting configuration*. The number

[5] Nodes that have no outgoing edges and that are not output nodes.

[6] We identify as *starting configuration* a triplet composed of the agent initial position and orientation and the light source initial position in the simulated environment.

of simulations `T` shouldn't be exhaustive but rather enough to assert that the network behaves adequately in most interesting configurations; the rest is left to the generalization capabilities of the network.

Furthermore, in order to minimize unnecessary score fluctuations due to spawning the agent too close or too far away from the source, it has been chosen that all the starting configurations must be characterized by the equal agent initial positions, same light source initial positions but variable agent rotations. The initial distance between the agent and the source must not be small, that is, it should be wide enough to fully employ the given simulation time to reach the target. Hence, *during training are rewarded more all those networks that successfully reach the light source.*

Each simulation scores s is then aggregated with the others to produce a global score vector (m, d), which components are `mean` and `standard deviation`:

$$(m, d) = \frac{\sum_{t=0}^{T} s_t}{T}, \sqrt{\frac{\sum_{t=0}^{T}(s_t - \overline{s})^2}{T}}$$

In order to explain why such statistics have been chosen, let us first list the possible examined alternatives. For simplicity lets consider only the the case of `phototaxis`:

- Get the worst value (`maximum`) among the simulation scores: given global score s, if a successive evaluation has overall better scoring but same worst case then it would be accepted as new solution for a *sideway move* but, at the same time, if a successive solution has overall worse scoring but better worst case[7] then that solution would also be chosen. This lead to an oscillating trend in the function overall scores that nullifies previously done work.
- Get the best value (`minimum`) among the simulation scores: similarly to the above explanation, the overall contribution is never taken in consideration thus leading to unbalanced networks where are rewarded those lucky enough to get a favorable *starting configuration.*
- Get the `median` value among the simulation scores: like for the worst case approach, if solution with same median but overall worse

[7]Remember that we are minimizing the score during `phototaxis`.

scoring is found, either higher Q1 or Q3, for the effect of *sideway moves* that solution would be chosen.

– The same would happen when using *only* the `mean` as global score, even if vaguely attenuated since the `mean` already gives an idea of the overall contribution of each score.

The usage of `standard deviation` samples only those *sideway moves* with overall better scoring, even when the `mean` scores are equal. That is, given solutions with equal `mean` scores, the `comparator` will look for those with single scores nearer to one another. Instead, for different `mean` scores, it will always look to the smaller one, independently from their `standard deviation`. One downside of this approach is a possible *over-training* of the network: this can be avoided by imposing an adequate target score that is neither too small nor too big.

Note that for `anti-phototaxis` the above reasoning still stands, obviously remembering that each scores should be maximized instead.

- `comparator` – As anticipated while describing `evaluator`, the evaluation score to be compared against the target score is a vector (m, d):

$$\texttt{compare}((m,d)_i,(m,d)^*) = \begin{cases} m_i < m^* & \text{if } m_i \neq m^* \\ d_i < d^* & \text{else} \end{cases}$$

In both cases *sideway moves* have been kept but handled in a different manner than in the original idea: instead of allowing all the solutions with scores equal to the current one, the solution are sampled through the `standard deviation` in order to get only those that effectively move towards an improvement of the solution. An interesting note, observed during the experiments, is that even if the score of new found solutions visibly fluctuates with each minimal changes in the network internal structure, the sub-space of new solution still appear plateau-ish since equivalent solutions are far from being rare. Therefore, *sideway moves* still work as intended in the original work of [16].

2.5.3 Testing Methodology

The testing of the `Behavioural-BN` exploits most of the methodologies previously described and is divided into two phases: *data collection* and *data analysis*. The main scope of the tests is to assert whether the development procedure is able to create capable networks most of the time.

Data Collection

The primary concerns in this phase is to execute `Z` test instances, each constituted by `T` simulations and paired to a random *starting configuration*. The collected simulation data (see Section 2.5.2) are aggregated into the following data:

- *agent score* – Is the main descriptor of the agent overall behaviour as described in Section 2.5.2.

- *agent initial y-axis rotation* and *agent initial distance* – They give an approximate estimate of the difficulty of applying the behaviour to the current task. For example: achieving `phototaxis` from near the light and oriented in its direction is far more easier than achieving the task from very far away with the agent front opposed to the source. The initial distance is given by the `euclidean 3D-distance` between the *agent position* and the *light source position*, both at the simulation step `0`.

- *agent final distance* – Necessary to corroborate the *agent score*: during `phototaxis`, the smaller the score, the smaller the final distance; in `anti-phototaxis`, the greater the score, the greater the final distance. The final distance is given by the `euclidean 3D-distance` between the *agent position* and the *source position*, both at the final step of simulation.

Along the *agent score*, in order be able to better reward worthy networks, another scoring model *ws* has been employed. It is however obtained by normalizing the *agent score* by the ratio between the *agent final distance* and the *agent initial distance*:

$$ws_{zt} = s_{zt} \cdot \frac{\texttt{fdist}_{zt}}{\texttt{idist}_{zt}}$$

where t is the number of the simulation inside the test instance z.

The idea is to reward even those network that starts from disadvantageous distances, regardless of the agent orientation, since they may be correctly achieving the target behaviour without being able to effectively reach the proximity of the light source or escape far away due to the limited time.

For example during `phototaxis`, since the score should be minimized, the ratio $\frac{\texttt{fdist}_{ik}}{\texttt{idist}_{ik}} \to 0$ as the agent correctly execute the behaviour from far away and moves closer to the source, hence further decreasing the score. This is especially significant for those networks that start moving from far away, while it remains quite unchanged for those who start already near or just slightly move from their initial position since $\frac{\texttt{fdist}_{ik}}{\texttt{idist}_{ik}} \to 1$. While executing `anti-phototaxis` the idea is similar but, since the score has to be maximized, the score will grow larger as the agent starts nearer from the source and gets further away: $\frac{\texttt{fdist}_{ik}}{\texttt{idist}_{ik}} \to \infty$. For those networks that instead already start from afar or remain suspended under the light, the score remain unchanged since $\frac{\texttt{fdist}_{ik}}{\texttt{idist}_{ik}} \to 1$.

This scoring approach differs greatly from the one used throughout development phase since, there, only knowledge internal to the agent could be used. Moreover, during development, all agents and light source were placed always at the same coordinates, while, in testing, the agent initial position can be whatever.

Data Analysis

During this phase the aggregated data are analyzed, ranked and plotted, grouping them together by either single test instances or by network, in order assert (A) how many capable networks are generated by the development procedure, (B) discretize the best networks among the many and (C) attest if there is any correlation among the successful networks and their parameters.

The statistics on the data of each test instance or network are summarized by the following plots:

- Box-plots of test score s and weighted scores ws – A concise way to express, for each test instance or network, the median, the first and third quartile, the lowest datum still within 1.5 `IQR` of the lower quartile (Q1) and the highest datum still within 1.5 `IQR` of the upper quartile (Q3), where `IQR` is the `Inter-Quartile Range`, given by the range $[Q1, Q3]$.

- Bars plots expressing success rates over multiple score thresholds by network or test instance. This plots are generated for each network based on the percentage of tests that achieved a score lower that proposed threshold. It is also useful to assert the exact ranges where scores are distributed.

- 2D Scatter-plots of scores and weighted scores distribution in relation to their initial or final distance from the source.
- 3D Scatter-plots of scores and weighted scores distribution in relation to their initial or final distance and orientation from the source.

The various plots are ordered following a simple ranking model, based on the sum of various rank scores. Each rank score is obtained by ordering[8] each network or test instance over the *average scores* and *average weighted scores*, then assigning to each of them a rank equal to their position in the sequence. That is, the lower the rank, the better it scores in the ranking. The total rank is given as a sum of the accumulated ranks on each descriptor, hence the network that scores the lowest sum of ranks is the overall best one.

2.6 Assembling the Robot Controller

Now that all the necessary components of the `Selective-BN-Controller` have been defined, it is time to assemble them together. In Section 2.3 we first defined the mapping between the two fundamental components, then respectively described them in Sections 2.4 and 2.5: `SBN` and `BBN`.

First, we will swiftly review how the various components interact among them; then move on, describing the testing methodology, starting from the data collected for the evaluation, and, finally, the evaluation method itself.

2.6.1 Connecting the Networks

Remember that the idea here is to unambiguously associate one or more attractors of the `SBN` and a precise behaviour, encapsulated in a `BBN`, to a common input stimuli. There, attractor states are assigned to their attractor while non-attraction states are mapped into other attractors following the transitions probabilities described into the *Attractor Transition Matrix* of the `SBN`. At boot time, the agent has even probabilities of being in any of its possible attractors.

When the `Selective-BN` capture an environmental signal on its input node, in absence of noise, it updates its internal state and, based on that, choose a behaviour by applying the above described mapping. Then sets on each input nodes of the chosen `BBN` the *binarized*[9] *irradiance* perceived through

[8] Ascending for `phototaxis` while descending for `anti-phototaxis`

[9] Since the network nodes have binary states, it follows that sensors values, whatever they represent, must be transformed to a binary value o string.

the available sensors. The `Behavioural-BN` will in turn updates its state and sends a signal to the actuators based on the state of their associated output nodes. The same happens when under the effect of *noise* but, on each step, the network may choose a different behaviour.

2.6.2 Testing Methodology

While we are testing the whole combination of networks, the real goal of the process is to assert (1) if the chosen mapping lead to a sound global behaviour and (2) whether or not the developed `Selective-BN` is able to exploit its properties to properly make use of the mapped behaviours. Since the `SBN` is solely generated to satisfy the specified constraint, a test in a simulated environment is required.

The methodology employed for testing the network is very similar to the one used to evaluate the `Behavioural-BN`. What has changed are, primarily, the aggregated data, which now exploit both minimization and maximization of the objective functions previously defined in Section 2.5.2, and the way the agent *starting configuration* is generated.

Target Behaviour

This time the test of the network involves both the behaviours of `phototaxis` and `anti-phototaxis` previously defined. Moreover, the network will be initially subject to *noise* until an environmental signal, simulating the *noise* subduing, is perceived by the agent and leaves it free to develop an adequate behaviour to cope with that perceived stimuli.

In order to simplify the evaluation and increase the probability for the agent to express both behaviours, the environment dynamics, during the agent simulation, have been subdivided into four phases: (1) *noise*, (2) `phototaxis`, (3) `anti-phototaxis` and, finally, (4) *noise*. That is, initially the robot must be able to differentiate from its pluri-potent state when the *noise* subdues, then re-differentiate to a new behaviour when the environmental stimuli changes and, finally, regress to its initial stem-like state when the *noise* returns.

In this case, the above phase subdivision has been chosen in order to avoid that `anti-phototaxis`, if expressed before `phototaxis`, took the agent too far away from the source, hence potentially unable to express the phototaxis behaviour. Furthermore, as stated in Section 2.3, it is possible that during *noise phase* some of the expressed behaviours have more influence on the global conduct than others, due to random fluctuations on the behaviour choice or an higher average speed of one of the behaviours. As such, the *noise phase* may already lead, erratically, the agent close or further away from the source.

Robot Simulation

Since the effectuated tests will involve a `Selective-BN`, new data are needed along those already collected for the connected `BBNs` (see Section 2.5.2).

These newly added information are:

- *current attractor* – Collecting an attractor id $\mathtt{a}_i$ is useful for a number of reasons: (1) corroborate that each behaviour is characterized by a unique set of attractors, (2) if there are other expressed behaviours[10] outside the wanted behaviour and (3) check whether during *noise phase* the attractors really exchange each other following the probabilities of the network *Attractor Transition Matrix*.

- *perceived environmental stimuli* – Knowing if and which stimuli the agent has captured, along with which attractors has been expressed, is useful to corroborate the wanted `constraint #4` that each stimuli should lead to only and only one behaviour.

- *noise active* – Useful for tagging each step of the simulation to whether or not it was influenced by noise.

Moreover, since the environmental stimuli is binary, `phototaxis` is always associated to stimuli 0 while `anti-phototaxis` to stimuli 1 but their associated attractor may differ on each `Selective-BN`, depending on which attractor has been associated to input `0` and which to input 1.

Also, in order to correctly discretize each phase from one another, they are all given a precise time duration.

Data Collection

Like in the previous testing, `Z` test instances, each of `T` simulations and random *starting configurations*, are executed during the *Data Collection* phase.

From the aggregation of the above proposed data and those previously defined in Section 2.5.2, some new descriptors are generated for each single simulation. Given that the number of *noise phases* are two and the number of wanted attractors is $M = 2$, since we have only `phototaxis` and `anti-phototaxis` to achieve, then:

- *noise* $\mathtt{k}$ *attractor* $\mathtt{i}$ *count* – For each *noise phase* $k \in \{1, 2\}$, the number of repetition of each attractor a_i, $i \in \{0, 1\}$.

[10] Remember that each state is associated to an attractor based on either is membership to the attractor or the transition probability expressed in the network `ATM`

Remember that a_0 is not necessarily associated to `phototaxis` and either is a_1 to `anti-phototaxis`, since each attractor is dynamically associated to an input stimuli which define the wanted behaviour:

$$\begin{cases} \texttt{phototaxis} & \text{if stimuli is 0} \\ \texttt{anti-phototaxis} & \text{if stimuli is 1} \\ \texttt{random} & \text{if no stimuli, } \textit{noise phase} \end{cases}$$

Moreover, a_0 is not the attractor associate to stimuli 0 but simply the first found attractor among those expressed. The same apply for a_1.

- For each `phototaxis` and `anti-phototaxis` phase in the simulation are aggregated:
  - *agent score* – As expressed from the objective function in previously defined Section 2.5.2.
  - *attractors* **`i`** *ratio* – The ratio of non-phase attractors over phase attractors:

$$r^0 = \frac{|a_i^0|}{|a_j^0|} \qquad\qquad r^1 = \frac{|a_j^1|}{|a_i^1|}$$

    where $i, j \in \{0, 1\}$, $i \neq j$, depending on the stimuli mapping of attractors and behaviours. The apex refers to the actual phase between `phototaxis` (0) and `anti-phototaxis` (1).

    That is, the ratio of attractor expressing `anti-phototaxis` during *phototaxis phase* and, the other way around, the ratio of attractors expressing `phototaxis` when in *anti-phototaxis phase*. Both these values should optimally tend towards 0 since we would expect that the number of respectively associated attractors exceed the others in their own phase.
  - *agent initial distance* and *orientation* – As previously stated they give an approximate estimate of the difficulty of applying the behaviour to the current task.
  - *agent final distance* and *orientation* – They corroborate the *agent score* since they both share the same trend: it decreases in `phototaxis` while it increments on `anti-phototaxis`.

From the aggregated data some more descriptors are further generated:

- *noise* **`k`** *attractors ratio* and *difference*

$$r_{01}^{noise_k} = \frac{|a_0|}{|a_1|} \qquad\qquad d_{01}^{noise_k} = |1 - r_{01}^{noise_k}|$$

  where k is the number of the *noise phase*. The first is the ratio which value should tend towards `1`, that is, the number of counted repetition for each attractors should be similar to one another. The latter descriptor, instead, expresses the absolute distance of the ratio from the balance point `1.0`. They are both computed for each *noise phase*.

- *noise* **`k`** *attractor* **`i`** *percentage* – The percentage of expressed attractors $a_i,\ i \in \{0, 1\}$ among all the expressed ones:

$$p_i^{noise_k} = \frac{|a_i|}{|a_0| + |a_1|}$$

  where k is the number of the *noise phase*. In this case it is determined for each *noise phase*.

- *weighted score* – Follow the same logic already explained in Section 2.5.3. It is computed for each `phototaxis` and `anti-phototaxis` phase.

- An ordered set of Boolean descriptors B that holds the satisfaction of the following constraints:

  1. $$|p_0^{noise_0} - \overline{atm}_0| < \tau \wedge |p_1^{noise_0} - \overline{atm}_1| < \tau$$

  2. $$|p_0^{noise_1} - \overline{atm}_0| < \tau \wedge |p_1^{noise_1} - \overline{atm}_1| < \tau$$

     The difference between the percentage $p_b^{noise_k}$ and the averaged `ATM` probabilities of those attractors associated to the same behaviour should be lesser than τ, during both the *noise* phases and for both behaviours. That is, the percentage of expressed attractors should closely match the mean of the described transition probability and τ is needed to calibrate such difference since averaging those `ATM` transition is an approximation.

3.
$$fdist^0 - idist^0 < \delta$$

4.
$$fdist^1 - idist^1 > \delta$$

During *phototaxis phase*, the difference between initial and final distance should be less than δ (centimeters), that is, the initial distance should be at least greater than final distance. δ is used to avoid that `phototaxis` `BBNs` that start too close from the source get penalized, as such, it should be calibrated accordingly. When expressing `anti-phototaxis`, the difference should be always greater than δ, therefore the initial distance should be lower than the final distance. This way we penalize those agents that only move less than δ centimeters away from the source.

5.
$$ws^0 < \sigma$$

6.
$$ws^1 > ws^0$$

The measure of the weighted `phototaxis` score should respectively be lower than an optimal `phototaxis` score σ, while the weighted `anti-phototaxis` score must be always greater than the `phototaxis` score computed for the same simulation.

7.
$$r^0 < \rho$$

8.
$$r^1 < \rho$$

Both the `phototaxis` and the `anti-phototaxis` attractors ratio should be lower to the same threshold ρ, that is, both should tends toward 0. Since during the *transition phase* may appear some random attractors that do not match the wanted behaviour, due to the imposition of the input stimuli on the network input nodes, the threshold ρ is used to add flexibility to the constraint else only few special networks would satisfy it.

Moreover, from the previous Boolean descriptors, two more are arranged: *satisfaction score* – ss and *satisfaction rate* – sr:

$$ss = \frac{\sum_i^{|B|} s_i}{|B|} \qquad sr = \begin{cases} 1 & \text{if } ss \equiv 1 \\ 0 & \text{else} \end{cases}$$

The former describes, since each of the previous descriptor is a Boolean, the percentage of satisfied constraints while the latter simply tags a simulation whether or not has satisfied all the constraints.

Data Analysis

Now that each data necessary to the analysis has been exposed, first we will swiftly provide a list of the employed statistic and plots, as previously done for the `Behavioural-BN` and then move to define the adopted ranking methodology:

- Box-plots of test scores s and weighted scores ws for both `phototaxis` and `anti-phototaxis`.
- Box-plots expressing the ratio between `anti-phototaxis` and `phototaxis` test scores s and weighted scores ws. This plot is useful to assert whether or not `anti-phototaxis` scores are always greater that their associated `phototaxis` scores, that is, their ratio is always greater than one.
- Bar-plots expressing *satisfaction score* and *satisfaction rate* by network or test instance.
- Bar-plots expressing the satisfaction of the boolean descriptors set B by network or test instance.
- Bar-plots expressing *attractor length* by network.
- 2D scatter-plots of scores and weighted scores distribution in relation to their initial or final distance from the source.
- 3D scatter-plots of scores and weighted scores distribution in relation to their initial or final distance and orientation from the source.

For what concern the ranking methodology, we opted for a rather simple one, based on the same concepts of the one already described in the previous section: each network or test instance has been ordered by the scores received

in some of the previously defined descriptors. These descriptors are: *weighted agent score* in both behaviours, *noise* `k` *attractors difference* in both phases, *satisfaction score* and *satisfaction rate*. The first four descriptors are used to represent the networks performance for each phase inside the ranking system, while the last 2 descriptors give a global view of the overall networks performance. Once the single score has been ordered atomically, each network or test instance receive a ranking score equal to their position in the sequence. If two elements have the same descriptor score they receive the same ranking score. The final rank is given by the sum of each obtained ranking score: the network or test instance that received the lesser score is ranked first and so on. In the end is rewarded the network or instance that has achieved overall good scores, without never going too low in a rank.

Expectations

What do we expect could characterize good a `SBN`? The characterizing features at our disposal on which we can leverage our assumptions primarily involve: the number and length of attractors, and transition probability between one another. A network with long attractors means that there are more attraction states mapped unambiguously to a specific behaviour hence reducing randomness in *noise* and transition phases, and, possibly, the length of transition phases between attractors. On the other hand, smaller attractor are sign of better generalization capabilities of a network [7], even if in our case they don't directly express a behaviour. Also, the more the attractors a network may express, the harder it becomes to effectively display all of them. Transition biases among attractors in the order of $\rho = \frac{1}{|A|}$ are optimal since they unconditionally help the network to achieve equally easier jumps from one attractor to another when a perturbation strikes. This also means that the agent is less likely to get trapped into the same single attractor when the agent is affected by noise.

In the end, we could say that good `Selective-BN` are characterized by relatively short attractors and transition biases as closer as possible to $\rho = \frac{1}{|A|}$. In our case the matter of the length of attractors is secondary since we look for a specific quantity of them.

Operatively speaking, a network of this kind should be able to freely express all its attractors, whatever their number may be, within noisy environments, while being capable of displaying and moving across all its behaviours whenever a stimuli is perceived or changes.

Conclusions We aim to automatically design and develop a robot controller based on Boolean networks that is able to both differentiate and then go back to its pluri-potent state or directly move to a new behaviour. In order to do so, we divided the design problem into 3 sub-problems: (A) design a Boolean network capable of re-differentiation that we called `Selective-BN`, (B) develop two or more networks that would implement one target behaviour each and that we called `Behavioural-BN` and then (C) map each attractor expressed by the `SBN` to a developed `BBN`. The development process of each network is tackled with a stochastic descent characterized by different level of granularity: `Generate-and-Test` and `pVNS`. Each developed network is then tested as a robot controller to assert their performances. The performance measures that guide both development and testing are based on the collection of the *irradiance* perceived from the agent sensors throughout its simulation.

In the next chapter we present the results of both the networks design or development, and their tests into a simulated enviroment. Also, the robot embodiment and the deployment environment will be described.

Chapter 3

Results

3.1 Experiments Set-Up

In this section we will illustrate and discuss the results obtained from the tests on the developed networks by employing the methodologies described in the previous Chapter 2. All the experiments are run with `Webots` [20] simulator and no real world deployment has been performed but is a possible future development.

First we present an overview on the robot model used, then a swift description of the deployment environment and finally the experiments alongside their results. The experiments will start with the analysis of `Selective-BNs` development process and their constraints satisfaction, then we move to the results regarding the `Behavioural-BNs` development and tests on both inquired behaviours: `phototaxis` and `anti-phototaxis`. Finally, we will analyze the results obtained by testing the complete `Selective-BN-Controller` on the robot.

3.1.1 Agent Embodiment

The adopted robot model is a simulated E-puck v2[1], visible in Figure 3.1, an improvement to the classic E-puck robot. It is a small differential-wheeled mobile robot designed for education and research. It has the same shape as its predecessor with great improvements and new features, such as: a WiFi module, a micro USB connector, a more powerful micro-controller, more memory, a Time-of-Flight distance sensor and an inertial measurement unit with a magnetometer. More information can be found in the link provided in the footnote.

[1] `https://www.cyberbotics.com/e-puck`

Sensors

We will briefly overview only those sensors equipped on E-puck which are useful to the means of the experiments, specifying characteristics like range, *noise* and, on the simulated side, the look-up-tables that characterize the sensor behaviour.

- `IR Sensor`: E-puck is equipped with eight `Infra-Red Sensors`, each of which works as both *ambient light receptor* and *object proximity meter* with a maximum sensing distance of 6 cm. In our study case, a slightly different version of the default look-up-tables have been used:

$$\begin{bmatrix} 0 & 0 & 0.01 \\ 1 & 1000 & 0.01 \end{bmatrix}$$

  where the first column is the perceived ***`irradiance`*** in $\frac{W}{m^2}$, the middle column express the mapped min and max float values, while the last column is the *noise* applied on the perceptions. The default value of sensor noise would have been 0.0, hence no errors in the sensors readings. The sensor resolution has been set to 10^{-5}.

- `GPS`: Differently to the original design, a GPS sensor has been added to the embodiment in order to track the position of the agent in the simulated environment. This sensor has no LUT and the default parameters have been left untouched; therefore the sensors always gives the exact position of the agent. This is obviously useful only in a simulated environment since the position of the agent is only used for a posteriori evaluations, therefore it didn't seemed necessary to give it a more realistic setup.

- `Bluetooth`: The Bluetooth module, given is generic nature, is employed to simulate *noise* and *environmental stimuli* as radio signals on the agent receptors. The noise and stimuli are associated as follow:

$$\begin{cases} -1 \rightarrow \textit{noise} \\ 0 \rightarrow \texttt{phototaxis} \\ 1 \rightarrow \texttt{anti-phototaxis} \end{cases}$$

  The sensor is adopted only as a receiver, using a single radio channel with a buffer of 4 Bytes.

Binarization Functions As previously mentioned in Section 2.6, sensors readings must be converted to binary before being applied on the input nodes of the network. The adopted binarization functions are rather simple:

$$x_i^{pt} = \begin{cases} 1 & \iff s_i > \zeta \\ 0 & \texttt{else} \end{cases}$$

$$x_i^{apt} = \begin{cases} 1 & \iff s_i \leq \zeta \\ 0 & \texttt{else} \end{cases}$$

where $\zeta = 0.0$ in our case scenario.

That is, during `phototaxis`, the function sets to 1 those node states where the sensed values are any greater than 0.0 while setting 0 on the others. In `anti-phototaxis` the function output is instead negated: it sets 1 where the readings are 0, otherwise 0.

Various other mapping were thought but this seemed the most valuable since, whatever the distance might be, the agent will always be able to perform its behaviour even with minimal stimuli from the environmental light.

Actuators

The employed E-puck actuators are:

- `Rotational Motors`: They are two, left and right, and can reach a linear speed between `[0.0, 7.536]` $\frac{cm}{s}$. In our study case, motors are employed only through binary speeds values since the output nodes of the developed network are always two:

$$wv_{side} \begin{cases} 1 \rightarrow \text{move} \\ 0 \rightarrow \text{stop} \end{cases}$$

  Moreover, with the employed coupling, the strength of the perceived stimuli doesn't influence in the power of the agent motion since each actuator is associated to only one output node and converted to a single binary value. However, it is possible to employ a more complex mapping where more output nodes are associated to the same actuator in order to express different speed levels but, in this case, how the perceived stimuli are associated and influence the speed values should be object of the training.

- `RGB Leds`: Differently from the original design, all the eight equipped leds on our epuck model are RGB and are employed during the `SBNC` experiments to better discern which behaviour the robot is expressing: `#ff00ff` – `magenta` for `anti-phototaxis` and `#00ff00`–`green` for `phototaxis`.

Morphology

The agent morphology, as seen in Figure 3.1, is the same of a standard E-puck model. Each pair of `Distance Sensor` and `Light Sensor` is positioned sequentially and clockwise along the perimeter of the robot at the following angles: $\frac{2}{5}\pi$, $\frac{\pi}{4}$, 0, $\frac{5}{3}\pi$, $\frac{7}{4}\pi$, π, $\frac{3}{4}\pi$, $\frac{3}{5}\pi$. In respect to the simulated world reference system, the internal model of the agent adopt its own, where the front side is positioned at $\frac{\pi}{2}$ while the 0 is found on the `Right Motor Wheel`. Hence, at π is positioned the `Left Motor Wheel`.

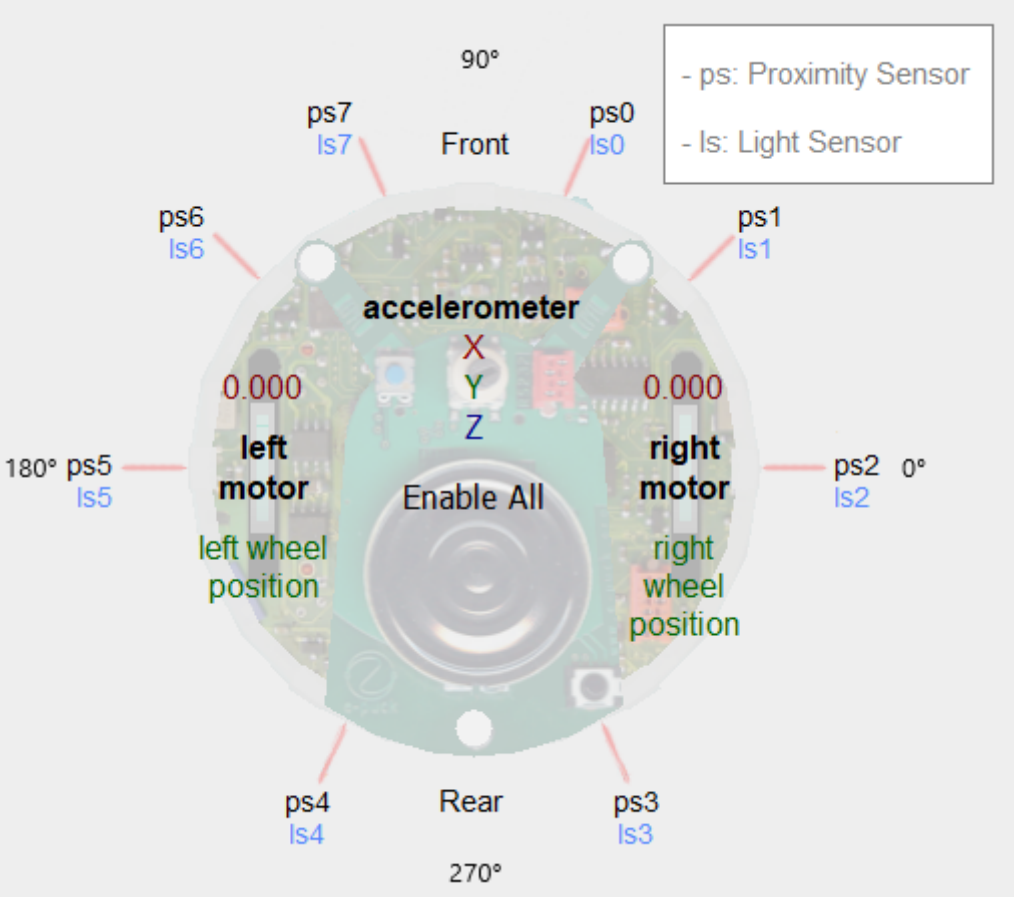

Figure 3.1: Agent morphology

3.1.2 Deployment Environment

The adopted simulation environment is rather simple as visible in Figure 3.2. The environment is composed of a square walled arena of size `N` × `N` with a `Point Light` of `Intensity` equal to `1.0` placed inside. The arena size and the position of both the light and the agent differs between development and tests.

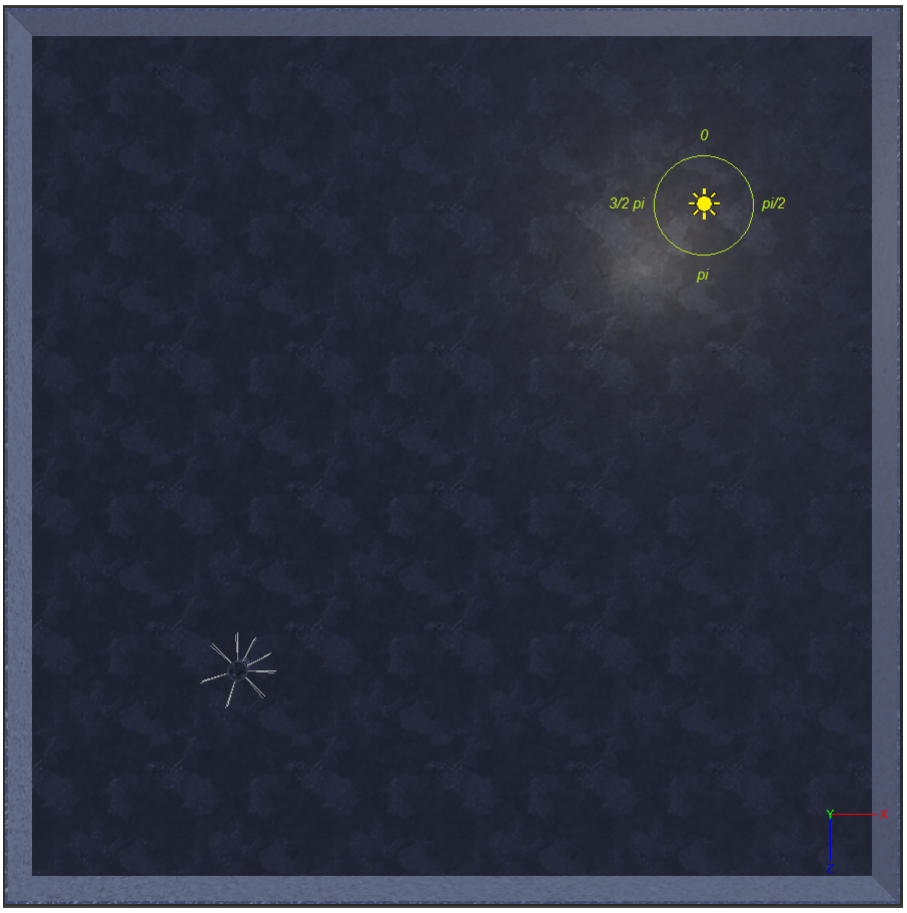

Figure 3.2: Development environment of a `Behavioural-BN`. In the simulated environment, the rotation of the agent body uses a different reference system from the one employed to describe the location of the robot internal components. As can be seen in the small reference system in the lower right corner, the y-axis points outward the figure, the z-axis is the vertical component while the x is the horizontal one. Rotations are distributed clock wise, as shown by the green unit circle around the light source, with angle 0 located along the negative branch of the z-axis. In this picture the agent is pointing the light source, therefore has a rotation of $\frac{pi}{4}$ radians or 45° degrees. In all our experiments we adopted the world reference system instead of the agent one.

During development of `Behavioural-BNs`, the arena is sized 3×3 with the light and the agent positioned in opposite corners, as shown in Figure 3.2, when expressing `phototaxis`, while they are positioned near one another when expressing `anti-phototaxis`. During the tests the light source is instead positioned at the center of a 5×5 arena with the agent randomly positioned at a distance between $[2.0, 2.5]$ meters around the light source or slightly under it, between $[0.0, 0.5]$ meters, depending on the behaviour to be expressed, respectively: `phototaxis` and `anti-phototaxis`.

The tests over the `Selective-BN-Controller` employ an similar setup to the last one described: a 5×5 arena with the agent randomly positioned at a distance of 2.0 meter from the light source. The distance is lesser than the previous tests for two main reasons: (1) the underlying Boolean networks used in the complete controller have already been tested and chosen accordingly to their performances, and (2) during the *noise phase* the agent may either close its distance to the source or get further away. As such, in order to increase the probability for the network to express both behaviours, the initial gap to the source has been reduced, this way even if driven further away it won't be enough to be unable to perceive any light.

This way, a good controller, since the `phototaxis` behaviour is expressed always before `anti-phototaxis` as aforementioned in Section 2.6.2, should first move toward the light after the *noise phase* subdues and, when the *anti-phototaxis phase* starts, be in proximity of the light source in order to successively step away from it. At the end a new *noise phase* will start, with the agent going back to its pluri-potent state.

3.2 Experiments and Results

In this section we will review the development and test experiments along with their results, starting from those involving the `Selective-BN`, then moving to the `Behavioural-BN` and in the end the tests over the deployment of the `Selective-BN-Controller`.

All the simulations have been executed on a Windows 10 Pro – Version 1903, Build 18362.295, mounting an `Intel i5 3570k 4/4 @ 4.2GHz` and `16 GB` of ram `DDR3 @ 1600MHz`. The `Webots` executable was loaded each time from a `WD Caviar Black 500GB @ 7200 RPM Sata 3`, while data were written by the simulator at the end of each simulation on a `WD Caviar Blue 500GB @ 7200 RPM Sata 2`. The registered average simulation times in batch mode are $7 - 8$ seconds for the `BBNs` development and tests, while 20 seconds for the `SBNC` tests.

For simplicity and in order to ease the reading, most of the Figures showing the statistical results of each experiments have been moved to the Appendices A, B, C and D at the end of this thesis. Each appendix respectively collects the Figures with the results regarding `SBNs` constraints satisfaction, `BBNs` expressing `phototaxis`, `BBNs` expressing `anti-phototaxis` and `SBNCs` deployment.

3.2.1 Selective Boolean Network

Network Development

The developed `Open Boolean Networks`, as previously stated in Section 2.1, are characterized by six control parameters: `N`, `K`, `P`, `Q`, `I` and `O`; while the constraints are defined by four parameters: ρ_{noise}, τ_{ij}, ϕ and `M`.

Before choosing which constraints parameters to employ, we explored the solution space of the Boolean networks that satisfy the target constraints. Therefore, for each set of network characteristics and constraints parameters, we generated 10000 random Boolean networks and evaluated on each one of them which constraints where satisfied. This approach let us assert which kind of network, identified by the same set of parameters, is more difficult to generate *already satisfying the given constraints*. That is, we looked for the sets of parameters with the higher probability of instantly producing a "perfect" network.

Constraints Satisfaction Statistics The network features and constraint parameters, used to generate the networks for this experiment, have been chosen in order to strike a good trade-off between solution space exploration and interested networks:

- For what concerns the networks, the chosen characterizing parameters are: `N` $\in \{5, 10\}$, `K` $\in \{2, 3\}$, `P` $\in \{0.5\}$, `Q` $\in \{0.0\}$, `I` $\in \{1\}$, `O` $\in \{0\}$. Those `N` and `K` have been chosen in order to explore small networks with different level of robustness [5]. Also, `I` and `O` are chosen since output nodes weren't necessary among the interested networks (moreover their number would not affect the solution), while there is need only of a single input node since the expressed behaviours are two, hence only two environmental stimuli are sufficient. The other parameters, instead, aren't particularly interesting for the constraints satisfaction nor the solution, therefore the default values have been assigned.

- `M` $\in \{2\}$ has been chosen since in our study case we have only two attractors but it is indeed suggested to extend the range if a wider understanding of the solution space is needed.

- $\tau \in \{0.19, 0.29, 0.39\}$ covers quite completely the ranges of possible interesting transition probabilities. Obviously if the network has more than five attractors, none of those network will satisfy the related constraint since we are looking for networks where each ATM entry ρ_{ij} is greater that τ, which already places an upper-bound equal to $\rho_{max} \sim \frac{1}{|A|} = 0.5$ in our scenario.

- $\rho \in \{0.1, 0.2\}$ should be a reasonable amount of *noise* to let the network express all the attractors without been trapped in them for too long nor being unable to express them at all.

- $\phi \in \{5, 10\}$ is simply chosen to be equal to `N`.

The number of explored solution is given by the combination of all the parameters and refers to a set of $4 \cdot 3 \cdot 2 \cdot 2 = 48$ different kinds of networks, each of which consists of 10000 randomly generated networks.

As we can see from the Figures[2] A.1, A.2, A.4 and A.5, the overall satisfaction for single constraints isn't hard to achieve, even if `constraint #4` is already quite hard to be satisfied. By analyzing the plots more thoroughly there are some interesting observations to be made:

- Overall, the network features N and K seem to greatly influence the trend of successful networks satisfying most of the constraints: the CSP becomes harder with increasing `N` and decreasing `K`. Indeed, both kinds of networks with $\texttt{N} \in \{5, 10\}$ are slightly more successful with $\texttt{K} = 3$ than with $\texttt{K} = 2$. Moreover, the difference in success rates between $\texttt{N} = 5$ and $\texttt{N} = 10$ is clearly not trivial since in some case becomes more than double for harder constraints.

- As for `constraint #1`, as seen in Figure A.1, it seems that networks with only two attractors occupy most of the solution space, with a success rate of $\sim$`0.38`.

- Contrarily to what previously thought, the `constraint #2`, as seen in Figure A.2, doesn't becomes harder as τ increases: the difference in the success rates is indeed noticeable between different networks but they always stand above ~ 0.2. On the other hand, a factor that may mislead the interpretation of this results are those networks with only one attractor which always satisfy the constraint since their only transition is $\rho_{00} = 1.0$. In fact, if we look at Figure A.3, we can see that by removing those networks the view greatly change, with a maximum success rate of 0.13 for networks with $\texttt{N} = 5$, $\texttt{K} = 3$ and $\tau = 0.19$.

[2]Networks are ordered by increasing N, τ and ϕ while decreasing `K` and ρ

- Strangely enough, in Figure A.4, ρ_{noise} doesn't seem to greatly affect the success rate of `constraint #3` but, from the figure, seems to amplify the difference between network features `N` and `K`. On the other hand, the adopted *noise* biases are pretty low; as such, for greater ρ values the difference may start to be more noticeable. Moreover, in our scenario the *noise* only affect a single randomly chosen node, hence its strength is limited.

- For `constraint #4`, shown in Figure A.5, the parameter ϕ seem to usually have only a small impact on the success rates, especially for networks with greater `K` due to their innate greater robustness since they are more chaotic. Moreover, it oscillates between values: some seems to satisfy the constraints better for $\phi = 10$ while others for $\phi = 5$.

- The joint satisfaction of the first three constraints in Figure A.6 leads to promising results with a mean success ratio of $\sim$`0.4` but, as seen in Figure A.7 which shows the addition of the last constraint, the rate drops to a tenth of that seen in the previous plot. Moreover, for some kinds of networks there aren't any successful at all, which is indeed unpromising since it seems to affect especially larger networks in the *critial regime*. But this is not strange news since the success rate of `constraint #4`, shown in Figure A.5, already addressed us towards the lower end of the straw.

In the end, for what we can deduce from these plots, especially the last, the problem seems indeed far from easy but we should also consider that it has been explored using only randomly generated networks. That is, if a more advanced algorithm, like `VNS`, should be employed then success rates would surely improve.

Therefore, having explored the desired solution space and made an idea on the distribution of the solutions, we have initialized the `Generate-and-Test` algorithm, employed to automatically design the `Selective-BNs`, with the following parameters: `max_iters` $= 1000$, `N` $\in \{5, 8, 10\}$, `K` $\in \{2, 3\}$, `P` $\in \{0.5\}$, `Q` $\in \{0.0\}$, `I` $\in \{1\}$, `O` $\in \{0\}$, `M` $\in \{2\}$, $\tau \in \{0.19, 0.29, 0.39\}$, $\rho \in \{0.1\}$ and $\phi \in \{10\}$. That is, each network has five, eight or ten nodes, one of which is an input and none are outputs, all with two or three predecessors each (including input nodes) and each truth-table entry as a probability of `0.5` to be true or false when generated. All nodes starts with their state set to false. All the transition probabilities ρ_{ij} of the networks are higher than the given threshold τ_{ij} and are able to move to another attractor after ϕ steps when a new stimuli is perceived.

Instead of generating only one kind of network and test a set those, we opted to generate one network for each of twenty-one different[3] kinds, each characterized by a different combinations of constraints parameters and network features.

Notice that we still kept $\tau \in \{0.19, 0.29\}$ since they don't necessarily lead to networks with low ρ_{ij} but that we also accept lower transitions probabilities among attractors, hence easing the generation of some networks.

3.2.2 Behavioural Boolean Network

In this section we will describe in detail the development and testing set-ups along with their results.

Network Development

The networks developed to become `Behavioural-BNs`, have been generated with the following characteristics: $\mathtt{N} = 20$, $\mathtt{K} \in \{2, 3\}$, $\mathtt{P} = 0.5$, $\mathtt{Q} = 0$, $\mathtt{I} = 8$ and $\mathtt{O} = 2$. That is, each network has twenty nodes, eight of which are inputs and two are outputs, all with two or three predecessors each (*except input nodes, which have none*) and each truth-table entry as a probability of 0.5 to be true or false when generated. All nodes starts with their state set to false.

Algorithm Set-Up The `pVNS` algorithm is initialized with the following parameters:

- `max_iters` $= 2000$
- `target` $= 4.0 \times 10^{-6}$
- `min_flips` $= 1$
- `max_flips` $= \mathtt{N} \cdot 2^{\mathtt{K}} - I - \mid n_{terminal} \in \mathtt{BN} \mid$
- `max_stalls` $= 100$
- `max_stagnation` $= 500$

[3] Where three out of twenty-one networks have been generated during preliminary phases with the same parameters but kept due to their interesting results.

Simulations Moreover, as anticipated in Section 2.5, each evaluation done by the algorithm is composed by $T = 6$ simulations, each 2 minutes[4] long and associated to a random *starting configuration*. Each configuration is characterized by the same *agent initial position*, $A = (-0.9, 0.0, 0.9)$, and *light initial position*, $L = (0.9, 0.0, -0.9)$, but different initial y-axis rotation (γ). These rotations are distributed in a hexagonal shape and sampled starting from $\gamma = \frac{\pi}{4}$ with an equal spacing of $\Delta\gamma = \frac{\pi}{3}$. This way the simulations will assert the most interesting scenarios with incremental difficulty: from agent facing the light, $\gamma = \frac{\pi}{4}$, until giving its back to the source when $\gamma = \frac{5}{4}\pi$, while exploring two in-between rotations on each side.

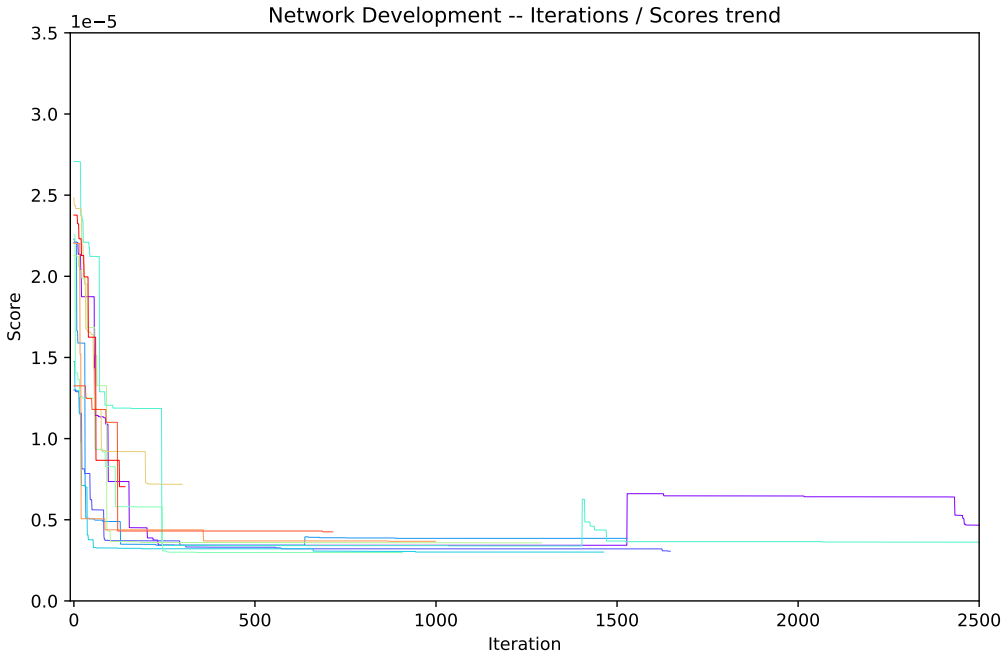

Figure 3.3: `BBNs phototaxis` scores trend by iteration during development. The proposed picture describes the evolution of the scores for each developed network as they are improved by the `pVNS` algorithm. Each step of the colored lines represents the discovery of a new and better solution; the only exception are those two steep increments around iteration 1500 which are related to networks that were placed again under development but with different aggregation strategies, from `median` to (`mean`, `stdev`), of the objective function score. Generally, we can see that the `pVNS` algorithm reaches saturation (0.5×10^{-6}) in less than 500 iteration, after which the network improvement becomes minimal. While 500 iterations seem few, remember that they imply 3000 robot simulations.

[4] $7 - 8$ seconds in batch mode

Network Testing

Next we move on the tests involving the development of the `Behavioural-BN`. This experiment is split into two separated parts: one concerning the development of a controller expressing `phototaxis` while the other a controller expressing `anti-phototaxis`.

For each controller type we will show the overall results of all the developed networks and an highlight on both a good and a bad network. As already stated, all the developed networks have $\texttt{N} = 20$, $\texttt{P} = 0.5$, $\texttt{Q} = 0.0$, $\texttt{I} = 8$ and $\texttt{O} = 2$. For what concerns the feature `K`, all the networks except one have been developed with $\texttt{K} = 3$ while only one[5] with $\texttt{K} = 2$.

Simulations Each network test is made of $Z = 10$ instances, each of which are composed of $T = 30$ simulations and associated to a random *starting configurations*. For each simulation composing a test instance, we decided to employ the same $T = 30$ *starting configurations* in order for the results to be comparable among one another. Each *starting configuration*, as anticipated in Section 2.5, is characterized by the same *light source position*, fixed at the center of the arena, and a random *agent initial position* within a radius $[2.0, 2.5]$ meters from the source; the *initial agent rotation* is chosen randomly without any kind of constraint.

In both the following sections we first present the achieved overall results, for which are presented the scores box-plots and success rates. Then one good and one bad network, which box-plots and success rates will be paired with scatter-plots displaying the score distribution in relation to agent *starting configuration*. All the data-sets, generated plots and ranking will be made available in a complementary material archive[6]. All the result are ranked left-to-right from the best to the worst network.

Phototaxis

Overall Results In the overall achieved score, as seen in Figures B.1 and B.2, each box-plot refers to single networks and is generated through all the data collected in the $Z \times T$ test simulations. From the first image, which displays the achieved `phototaxis` score solely based on light sensor readings, we can assert that only the first four networks among the ten developed can be labeled as really successful since their scores, outliers included, stands below

[5] Network ID `20190814T192337115`

[6] `https://github.com/XanderC94/BoolNetController/releases/download/0.0/data.7z`

$\sigma = 0.5 \times 10^{-5}$. The successive five networks, instead, are characterized by many outliers located in a gray area between $[0.5, 1.0] \times 10^{-5}$: most of their box-plot higher and lower whiskers are located around σ but their outliers far exceed the optimal threshold, especially network `20190714T104233346`.

The value σ is an empirical estimation, obtained after a process of trial and error during the initial stages of networks development. Indeed, σ is strictly related to the allotted simulation time of 2 minutes. But, depending on the development context, it may either represents networks that have expressed `phototaxis` or `anti-phototaxis`. In fact, with the given simulation time of 2 minutes, the agents are likely to score similar in both experiments: in the former has to reach the light starting from far away, while in the latter it has to escape from the light starting from its proximity. That is, supposing the networks have similar overall speed in both behaviours, the covered distance will be similar; hence they will be exposed to the similar amount of light radiations with peaks located on different time instants. Therefore, the time given to the agent to execute the task balances the scores differences since it indirectly defines the time spent by the network under the light. Indeed, the period spent under the light source is supposed to be higher during `phototaxis` while lower, but not none, on `anti-phototaxis`; surely, for higher simulation times the scores will start to diverge more clearly. The `phototaxis` score will continue to decrease while the `anti-phototaxis` one will end is growth as the light stops to be perceived.

The above described issue is unavoidable if we want a good trade-off between algorithm speed and completeness of behaviour evaluation, yet it is resolved by the adoption of the second scoring model, where both initial and final distance from the light source are taken into account, differentiating quite clearly the behaviours scores. When using the weighted scoring model the threshold σ is unchanged and neatly separates those network expressing `phototaxis`, $\sigma \leq 0.5 \times 10^{-5}$, from those expressing `anti-phototaxis`, $\sigma > 0.5 \times 10^{-5}$. To be more strict, good `anti-phototaxis` results are expected to be even above $\sigma \geq 1.0 \times 10^{-5}$, with the range $[0.5, 1.0] \times 10^{-5}$ as gray area where outlier networks are placed. Indeed, σ greatly depends, firstly, on the allotted simulation time and, secondarily, on the initial distance; hence $\sigma = 0.5 \times 10^{-5}$ is optimal only for our scenario.

If we look at Figure B.2, each network weighted performances stand out more clearly than before: all the four top networks are still situated in the optimal range between $(0.0, 0.5) \times 10^{-5}$, while most of those in the gray are now standing under the threshold σ, with the sole exception of `20190714T104233346` which outliers are still spread out. The only network[7] with $\mathtt{K} = 2$ stands on a

[7]Network ID `20190814T192337115`

edge: while displaying always a weighted score below σ, its outliers are spread around the whole range. This may be due to a slower overall speed of the agent which penalize the network on long distances. In fact, looking at Figure B.4, we see that, as the initial distance grows, the score grows more rapidly and spread wider.

What characterize both box-plots of those four top networks is a low variance in the first and third quartile, which become almost nonexistent on the weighted scores. This means that there is invariance when changing rotation and, to some extent, the distance from the light source.

The Figure B.3, where success rates on different scores thresholds are plotted as bars, shows us more clearly that most ($\geq 90\%$) of the scores in good networks are located between $[2.0, 3.0] \times 10^{-6}$, while roughly half of them is under the lower end of the range 2.0×10^{-6}. For what concern those borderline good networks, their score is under 3.0×10^{-6} most of the time as well ($\geq 80\%$). Therefore, if we would be more strict with the optimal threshold σ, another good value (only for `phototaxis`) would be 3.0×10^{-6}.

A Good Network Scenario Looking at the details and characteristics of one good experiment, we examine network `20190709T100603554_336`, which is also the best one. As can be seen in the Figures B.5 and B.6, each box represents the data of a single test instance and each composed of thirty simulations. It is noticeable that each box is characterized by a rather small `IQR`, as already could be seen in the global results. Moreover, the size of the `Inter-Quartile Range` seems to remain quite constant among all ten instances, with very few outliers, which show a constancy in the agent behaviour even on longer distances. This is further corroborated in Figures B.8 and B.9, showing respectively 2D and 3D scatter-plots, where the constancy of the behaviour performance is reflected in both plots through a plateau of scores which only slightly rise as the distance grows or the initial rotation changes and remains invariant to the agent initial rotation. Moreover, in both the scatter-plots there aren't evident outliers at all.

Lastly, in the middle Figure B.7 where the *success rate* is shown over various optimal thresholds, most of the scores ($> 90\%$) always stand between $[2.0, 3.0] \times 10^{-6}$ without any great oscillations. This further prove the constancy of the expressed behaviour.

A Bad Network Scenario Moving now to a worse scenario, we pick network `20190713T080716669` for the study. The first difference that strikes between the two networks is the inconstancy in both the normal and weighted scores of the worst one, as seen in Figures B.10 and B.11. The `Inter-Quartile Range` is, indeed, stretched towards the higher whisker in most of the test in-

stances. This difference can be further verified by looking at Figure B.12. In this plot is easy to see that only in some of its best instances the score is under 0.5×10^{-5} in less than the $\sim 80\%$ of the tests; that is, in the remaining $\sim 20\%$ simulations the network score is either in the gray area or even above 1.0×10^{-5}. In those cases, the agent probably just stood still or just slightly moved toward the light and then halted. Why so?

If we look at the Figure B.13 we can clearly see that the scatter-plot outline two distinct trends: one characterized by good scores and a slow growth, the other by not-so-good scores and, above all, a far greater growth rate. In Figure B.14 the previous plot is extended on a third dimension, there represented by the rotation along the y-axis of the robot body. It can be clearly seen that the plateau of points is split around the initial rotations between $[100°, 200°]$ degrees. This rotations refers to a *starting configuration* where the agent front-side is slightly pointing to the south wall of the arena while its side facing the light source is either the left one, for positive z components, or the right one, for negative z components. In the first case the agent is also giving its back to the source while in the latter the source is mainly found in front of it. This difference is probably symptomatic of the developed network which is unable to fully or optimally employ the agent embodiment on one of the two sides. By looking only at the scatter-plot, the first possible assumption is a deficiency in the network topology where some input nodes are connected only to terminal nodes, either single nodes or a cyclic sequence, hence wasting the contribution of some perceptions.

From the topology of the network shown in Figure 3.4, while we can see that node `17` is the only terminal node, by tracing all possible paths from one input node to each output node, we find out that all the input nodes are able to pass down their own contribution to at least one output node. That is, the network fully employ all the sensors contributions but is unable optimally use them. The reason maybe either being attributed to the Boolean functions that characterize each node or by both network topology and internal structure. Still, the development of this network ended at `mean` $= 4.585 \times 10^{-6}$ with a `stdev` $= 3.26 \times 10^{-6}$ due to `stagnation` at iteration `1532`. This means that it probably got trapped on some kind of local minimum and, even with *sideway moves*, was unable to get out of it before the stagnation mechanism activated. Another run of the development algorithm, using the actual network as initial solution, may probably increase its performances for the better.

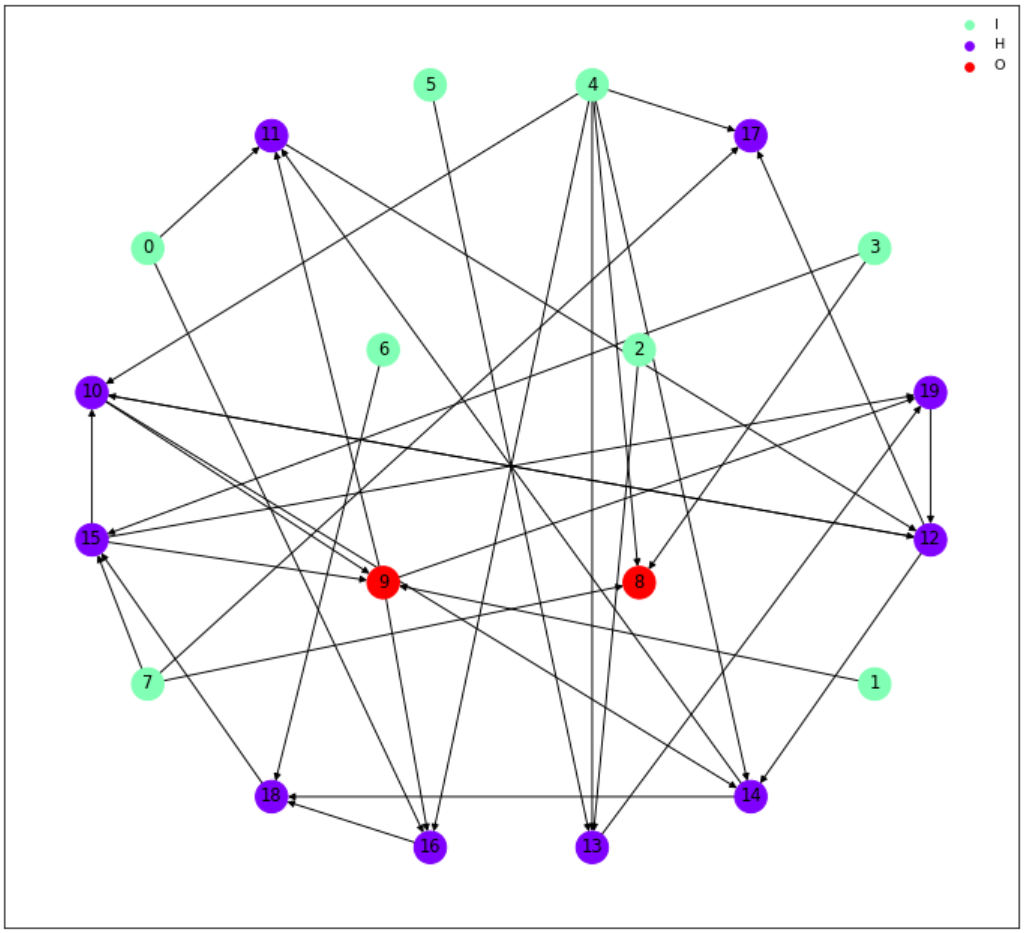

Figure 3.4: Network `20190713T080716669` topology.

Anti-Phototaxis

In the previous Chapter 2, we explained the whole process regarding the development of the `anti-phototaxis` behaviour and how it differs from the development of the `phototaxis` behaviour only in the maximization of the objective function instead of minimizing it.

While that is indeed the correct approach to the development, we thought it would be interesting to explore how the already developed networks to express `phototaxis` would behave with `anti-phototaxis`, exploiting the correlation between the two complementary behaviours. Therefore we tested the previously defined networks, changing only the *binarization strategy* to the one designed for `anti-phototaxis`.

Overall Results In the previous paragraphs, we pointed out how both `phototaxis` and `anti-phototaxis` score similarly in both their development and tests. As can be seen in Figure C.1 the scores strike great similarity with those achieved in the `phototaxis` tests. Since we are using the networks developed for expressing `phototaxis` this may lead to think that, even by changing the binarization function, the network still express the same behaviour. That is, it goes to the same *conditional attractors* [3], hence still expressing `phototaxis`, which is quite unlikely but indeed possible nonetheless. But, if we move to Figure C.2, it shows that this is not the case: the first, second, fourth and fifth networks attain very good results, always located above the previously defined threshold $\sigma = 0.5 \times 10^{-5}$. The third and the fourth networks, while ranked in good positions, display some spreading in their weighted scores. Similarly happens to the seventh and eighth networks. To corroborates these results, Figure C.3 shows the final distances achieved by each network along all their test instances: all said networks reach a final distance 1.5 meters starting from an initial distance from the source between $[0.0, 0.5]$ meters. Moreover, from the last figure we can see that all the outliers scores are associated to the same final distances, which are represented as outliers as well. It's interesting, though, how this outliers are associated to the distance of 0.3 meters which is the minimal distance from the light source the agent can achieve since the source itself is elevated by 0.3 meters from the ground: that is, the agent remained stuck under the light on some occasions.

On Figure C.4 are shown the success rates, subdivided by threshold, related to the weighted scores. We adopt this kind of plot since the use of the sole *anti-phototaxis* score won't let us grasp the real distribution of the achieved scores. As can be seen, most of the networks that correctly express the behaviour, achieved scores located for the most part ($\geq 80\%$) above 1.0×10^{-5} while the remaining ($\leq 20\%$) between $[0.75, 1.0] \times 10^{-5}$. Indeed these last scores are in an area we previously defined as "gray" but, as the Figure C.3 can prove, they are not associated to simulations that got stuck under the light, those have scores far lower than $\sigma = 0.5 \times 10^{-5}$ as clearly seen in the third, fourth and seventh networks in Figure C.4.

The results are indeed good but not something new: in the article by Roli et al. [6] they already obtained an agent capable of expressing both behaviour from a single network. What is surprising here is that these network aren't developed to express *also anti-phototaxis*. The behaviour is therefore emergent even if, in order to let network express it, is necessary to tackle with the binarization function. But why does it work? The reason can be found in the article [3] by Roli et al: by clamping the states of some nodes we are forcing the network to express *conditional attractors*. This attractors are characterized by patterns where some node states remain steady and immutable until another

clamping occurs, which move the network to another *conditional attractors*. Since the employed binarization functions are the negation of one another, they will indeed explore similar if not equal *input patterns* (on absolute terms), which translate in equal *conditional attractor* space, but they will activate opposed yet complementary attractors given the same environmental situation. We suppose that good solutions may be those that, for complementary input patterns, are capable to display attractors different from those exhibited during `phototaxis` and lead to an opposed behaviour.

A Good Network Scenario We now take a look to one of the good experiments: network `20190709T100603554_336`, which is also the one previously chosen for `phototaxis`. As shown in Figures C.6 and C.8, all the weighted scores are bounded between $(0.75, 2.0) \times 10^{-5}$ which is inside the expectations from the previous analysis and always higher than σ. On the other end the resulting weighted scores are spread on a wider range than the normal ones. If we look at the growth of the scores in Figures C.9 and C.10, we see that for normal scores is rather mild and linear[8] which is what we would expect: if the agent starts near to the source, then the *irradiance* is perceived more strongly and for more time, hence reducing the score, while if it starts already from a further distance, then the resulting score is obviously higher due to the overall lower irradiation received over time. In the second Figure, where the growth of weighted score is plotted, the spread of the score in the space has become wider, as could already be seen in the box-plots, even at similar initial distances. Even if we add a third dimension to the scatter-plot represented by the initial rotation, as shown in Figures C.11 and C.12, we already see in the first plot that the initial rotation holds no great information since the distribution of the scores is quite flat. In the second image, the weighting brings no higher understanding and the weighted scores result as well spread along z-axis even for similar initial rotations. This means that, for similar initial rotation and distance, it is possible to achieve far different final distances as shown from Figure C.13. The reason is probably due to where the agent is placed on the arena, since the agent is generated in the arena *around* the light source at a distance between $[0.0, 0.5]$ meters. Given that the agents behaviour depends on many *conditional attractors*, different position of the agent around the light source means different incidence of the light rays on the agent embodiment, hence different sensor reading and input patterns. Indeed this patterns are complementary between two opposed positions with the same initial rotation; therefore behaviours may be different. That is, for two opposed position around the light, characterized by same distance from it and same rotation,

[8] But still far different from the `phototaxis` one for the same network in Figure B.8

the achieved weighted score will be lower for those networks facing the light since they need time to rotate themselves, therefore reaching overall lower final distances. Only after rotating they will escape. Moreover, the fact that they need time to rotate themselves means that they are exposed for more time to higher level of light radiations, hence slightly decreasing their score.

A Bad Network Scenario Since there is only one evident failure in the proposed experiments, lets review the results of the network `20190710T020222266`. In the overall results, its weighted score box-plot covered the whole range of values $(0.0, 2.0) \times 10^{-5}$ which is translated, in term of distances, in $(0.25, 2.10)$ meters, rounded up. This scores certainly define a network that is unable to fully express the wanted latent behaviour. A characterizing elements of this network results, as shown in Figure C.14 and C.15, is that even after weighting the scores by $\frac{\texttt{fdist}}{\texttt{idist}}$, the median of the box-plot is always low and almost unmoved by the weighting. This probably means that both final and initial distance remains similar in values, hence that the network remain locked under or close to the light source. To endorse this claims, lets see how the final distance box-plots fare: looking at Figure C.16 we can see that, while there are some values that could be associated to a final distance of 0.3 meters, hence to the agents being stuck under the light source, the median stands always near 0.5; therefore the agent is more likely to just stand still or move around the light for some input signal configurations instead of escaping away. Moreover, Figure C.17 shows us that the majority of the achieved scores ($> 60\%$), in the best scenario, are below σ. That is, the third quartile contains less that the 40% of the achieved scores, distributed over a wider range of values of which Figure C.17 gives us a rough estimate, with most of them ($< 25\%$ of the total) just over 1.0×10^{-5}. This means that, in more that half of the simulations, the network has been unable to express the expected behaviour.

3.2.3 Selective Boolean Network Controller

Simulations The tests over the complete controller, composed of a single `Selective-BN` and two `Behavioural-BN`, are defined by $Z = 10$ instances, each composed of $T = 30$ simulations and *starting configurations*. For each network inside a test instance, the same $T = 30$ *starting configurations* are adopted in order for the results to be comparable among one another. Just like the tests on the `Behavioural-BN`.

What really differs is the simulation time, which is extended to 8 minutes[9] each and subdivided into four phases 2 minutes long. In order to simplify the

[9]Roughly 20 seconds in batch mode on the available hardware.

evaluation and data aggregation, as anticipated in Section 2.6.2, the first and the last phases are characterized by *noise*, while the two middle phases are respectively associated to a different binary signal that induce the network to differentiate, first to express *phototaxis* (0) and then, as the stimuli changes, *anti-phototaxis* (1).

The subduing of *noise* and the change of environmental stimuli are simulated through radio signals that the agent receives and consequently interprets as follow:

$$\begin{cases} -1 \rightarrow noise \\ 0 \rightarrow \texttt{phototaxis} \\ 1 \rightarrow \texttt{anti-phototaxis} \end{cases}$$

Remember that in the environmental set-up in composed by a square arena of size 5×5, the light source fixed at the center with an elevation of 0.3 meters and the agent is generated around the source at a distance of 2.0 meters.

The `BBNs` chosen to be paired with the twenty-one generated `SBNs` are those that resulted to be the best in achieving the target behaviours: network `20190714T214719298` for expressing `anti-phototaxis` and, to express `phototaxis`, network `20190709T100603554_336`. The reason to choose this two `BBNs` is to minimize the influence of the expressed behaviours on the performances of the developed `SBNs`, hence in order to correlate their differences solely and exclusively to their own intrinsic characteristics, like topology, Boolean functions, number of function inputs `K` and number of nodes `N`.

Overall Results Starting from the overall results obtained by the twenty-one different kinds of networks, in Figure D.1[10] the average *satisfaction rate* and *score* achieved by each network are represented. This values, as explained in Section 2.6.2, are aggregated from the satisfaction of the eight constraints described in the same section and which characterize the overall goodness of each `SBN`, along with their weighted scores in both behaviours and the descriptor for each *noise phase*. From the image we can see that, while networks are ordered top-down by their own rank, the bars are not in a perfect decreasing trend of heights but slightly fluctuates, like the fifth network from above. The same can be seen in Figure D.2, where the single Boolean constraints average satisfactions are plotted as bars. There, is also interesting to

[10]Notice that the label of each network in any plot is structured like {`network_id`}_{`N`}_{`K`}_{ρ_{00}}_{ρ_{11}}, where ρ_{00} and ρ_{11} are elements composing the main diagonal of the network `ATM`.

notice that networks are more likely to fail the first two constraints, which describe the average attractors expression during *noise phase*, and the fifth constraint, which tests whether the weighted phototaxis score is below the optimal threshold σ. On the other hand, the fact that the successive sixth constraint, testing that the weighted `anti-phototaxis` score is greater than the weighted `phototaxis` score in each simulation, has an average satisfaction rate in all the networks always greater than 90%, let us think that these networks indeed correctly achieve `phototaxis` and `anti-phototaxis` but they are likely to never get near enough to the light source (even for the weighted scoring model) to score below σ. This statement is further investigated in Figure D.3, where the ratio between weighted `anti-phototaxis` and `phototaxis` scores is shown: we suppose that, since `anti-phototaxis` score should always be greater than `phototaxis`, then their ratio should always be greater than 1.0. In the image, the first eleven networks always score above 1.0 with a lowest ratio of approximately 5.0, while most of the others even reach a ratio of 1.0. We also suppose that those network characterized by low ratio are those which score bad in the fifth constraint yet good in the sixth. Obviously the lower the ratio, the more likely the networks are to score worse in the fifth constraint: in order to have a lower ratio the network should either (A) express correctly `phototaxis` but get stuck under the light during `anti-phototaxis`, or (B) execute `phototaxis` from far away, which reflect in a higher `phototaxis` score, and then correctly express `anti-phototaxis` but from far away. The former case is not what we are looking for, since those networks would easy achieve a weighted `phototaxis` score below σ. Moreover, their weighted `anti-phototaxis` score wouldn't satisfy the sixth constraint since it would result similar to the `phototaxis` score due to the contribution from *initial* and *final distance* being trivial. The latter case, instead, while the `anti-phototaxis` weighted reward would be trivial since the network already starts from afar and therefore already higher than the weighted `phototaxis` score and σ, `phototaxis` score may be higher than σ even if the contribution from *initial* and *final distance* is not trivial. That is, these network satisfy the sixth constraint but not the fifth.

As previously stated, we expect that the *noise phase* may either let the agent get near, away or remain at constant distance from the source, mainly depending on the attractors transition probabilities of the paired `Selective-BN`. This means that, if an `SBN` has unbalanced probabilities which are far from $\tau_{ij} = 0.5$, networks are more likely to get too much near the light or too far away: the first case is associated to a predominance of the average attractors transition probabilities associated to the `phototaxis` stimuli, while the latter to a predominance of the average attractors transition probabilities associated to the `anti-phototaxis` stimuli. Therefore, with the chosen constraints,

unbalanced networks are penalized. This reflects, indirectly, in the fact that networks which fail in the fifth and sixth constraints are those brought far away during *noise phase* and therefore achieve a bad `phototaxis` score.

For what concern the failure of the first two constraints, the reason is difficult to point out since, even if transition probabilities are unbalanced, on average the network should spend the same amount of time on both behaviours *in the long term*. This means that either the time given to each *noise phase* is too short for the network to effectively and equally express both behaviours[11] or networks have been unfortunate, that is, they effectively expressed on both behaviours but then moved back on the other after a short while. The most probable among the two is surely the first and to corroborate this is the fact that all the networks score very similar in each *noise phase*: it would be difficult to be unfortunate in both *noise phases* and in all the networks simulations. Remember that during *noise phase* each behaviour is expressed following the transition probabilities displayed by the network `ATM`.

Instead of showing the normal `phototaxis` and `anti-phototaxis` scores we directly move to the analysis of the weighted ones, since the comparison of normal scores, as already stated, wouldn't be much explanatory due to scores being similar.

We first analyze the Figure D.4, where the weighted `phototaxis` scores are represented through box-plots. We can see that performances greatly vary between each network, even if the underlying network implementing the behaviour is the same. Roughly all the networks `IQR` remain below σ: fifteen of them with only some outliers between $[0.5, 1.0] \times 10^{-5}$, while the other six extend their whiskers above σ but always standing below 1.0×10^{-5} with only some outliers above.

Given that this time the initial distance from where the agent starts expressing phototaxis may variate greatly due to the erratic behaviour during the *noise phase*, all the networks seems to be scoring pretty well. Indeed, if we take a look at Figure D.6 where the weighted scores are plotted in relation to their initial distance, the weighted scores seem to mainly follow the trend already seen in Figure B.6. The sole exception is a cluster of weighted scores, characterized by higher scores and faster growth, neatly separated from the global trend. Since this plot doesn't give us more information it is necessary to extend it over an extra dimension as seen in Figure D.7, where the third dimension is represented by the initial agent rotation around the y-axis. The image shows that the cluster of outliers is located between rotations $[200, 250]°$, that is, when the agent has its front pointing the bottom edge of the arena,

[11] Remember that `SBNs` are designed to only express all their attractors after i updates, without any limitation on their percentage.

facing the light for negative z components, while giving its back for positive z components. But why are these so neatly separated while there are still better scores that are achieved with the same initial rotations? Moreover, the cluster is found only for scores achieved by exactly starting from a distance greater than ~ 2.5 meters from the source. Again this may be partially understood by taking in consideration the fact that the agent starts its `phototaxis` behaviour somewhere *around* the light source, therefore similar distances with similar rotations but opposed positions from the light may lead to different scores.

Moving now to Figure D.5, we'll analyze the `anti-phototaxis` weighted scores achieved by each network. As shown by the image, all the networks scores above σ and above 1.0×10^{-5} which is the stricter optimal threshold previously thought in Section 3.2.2. Furthermore many networks reach even higher scores but notice that these are the same networks that score "worse" on `phototaxis`: this means that, during *noise phase*, these network are taken further away from the light source, therefore they get higher scores when expressing `phototaxis` which become even higher while expressing `anti-phototaxis` since they start already from very far away. Indeed, looking at Figure D.8, we can see that many networks are induced to express the `anti-phototaxis` behaviour at a distance of 1.0 meter from the light source. This reflects the fact that these network also stopped expressing `phototaxis` when still far away from the light source.

Conclusions In the end, what greatly distinguish the performances of these agents is really and solely the underlying `SBN`. Those that perform well are paired with `SBNs` characterized by a good balance between the attractors transition probabilities that stand around $[0.4, 0.6]$. Moreover the results appear to be independent of the number of nodes `N` and number of function inputs `K`. The chosen noise bias $\rho = 0.1$ has proven to be high enough to let the network effectively and fully express both behaviours, even if not always homogeneously, during both *noise phases*. Furthermore, if we look at Figure D.9 we can also see that the length of the attractors doesn't provide any particular advantage since many good networks are characterized by short attractors while other by longer ones, without any clear trend. Indeed, the adopted sample of networks here discussed isn't wide enough to describe perfectly a whole family of possible solution but may still holds some significance.

Conclusions and Future Work

In this thesis we devised two automatic procedures, both based on stochastic descent but characterized by different levels of abstraction, which are able to design and develop effectively two different components of a BN-robot controller. The first component, called `Selective-BN`, is a Boolean network capable of controlled re-differentiation: to both differentiate and then go back to its pluri-potent state or directly move to a new behaviour. Its design problem is modeled as a Constraint Satisfaction Problem and solved through a `Generate-and-Test` algorithm which iteratively produces a new solution whenever the previous one does not satisfy the chosen constraints (see Section 2.4). The other component, called `Behavioural-BN`, is a Boolean network precisely developed to express a target behaviour among `phototaxis` or `anti-phototaxis`. Its development problem has been tackled through a `Variable Neighbourhood Search` algorithm which iteratively improve the initial given solution through a random exploration of the neighborhood of the incumbent solution. The search algorithm looks for networks that, depending on the behaviour to be achieved, either minimize or maximize each contribution obtained through an objective function (see Section 2.5.2) which, in our scenario, aggregates the *irradiance* perceived by the agent sensors at each simulation step. Moreover, in order to complete the controller, the two developed components must be connected; therefore we also realized a mapping strategy that indirectly associate attractors and behaviours. We dynamically associate each attractor expressed by the `SBN` to a precise input stimuli, while each `BBN` is instead paired a priori with one of the stimuli. That is, we dynamically and unambiguously associated attractors to behaviours.

Every `BBN` has been tested to assess its performance. We have chosen one good network for each target behaviour to be paired with the designed `SBN`. These well behaving networks have been selected in order to minimize the influence of the expressed behaviours on the performances of the developed `SBNs`, hence in order to correlate their differences solely and exclusively to their own intrinsic features. Finally, each `Selective BN Controller` has been tested as well in a similar way to the `BBNs`.

The obtained results are indeed promising with seven out of ten (70%)

developed BBNs capable of executing phototaxis and four out of ten (40%) networks able to correctly express anti-phototaxis. Among the twenty-one developed SBNs only fourteen (67%) have been able to score good in both behaviours while the remaining networks, even if capable of expressing both behaviours, where characterized by unbalanced *noise phases* that were likely to drive the agent far away from the light source. On the other hand, many networks, even among the good ones, scored poorly in the ranking constraints concerning the ratio of expressed behaviour during *noise phases*: only six where able to match the expected attractor expression ratio in over 80% of their simulations.

Also we haven't found a clear correlation between good SBNs and their features, if not for being characterized by overall balanced transition probabilities $\rho_{ij} \in [0.4, 0.6]$ between their expressed attractors. The number of nodes and number of inputs don't seem to exert any influence on their performances as well. Moreover, it would be interesting to increase the *noise* parameter ρ to values greater than 0.1 and study how the networks behave during those *noise phases*.

We also performed some experiments to test the solution space characterizing the problem of designing an SBN. The results of such experiments asserted that the problem is far from easy and becomes more complex as the network grows in the number of nodes but easier as the number of inputs to each node increase. Obviously, the more we look for network with balanced transition probabilities, the more difficult the problem becomes since we cut off many, yet unbalanced, solutions.

Furthermore, in four out of ten (40%) networks, we have been able to effectively exploit the correlation between two complementary behaviours with really promising results: instead of developing a complete and new network to express a new behaviour, we tested whether it was possible for networks solely developed to correctly express phototaxis, to also correctly express anti-phototaxis without them being purposely optimized for it.

Future Work The possibilities on future work opened by this thesis go from further investigating the results we discussed here, to the improvement of the employed design methodologies. In the latter case, an interesting development consists in tackling the SBN design problem with an SVN algorithm instead of a Generate-and-Test approach; this way good properties of generated networks will be kept instead of being completely discarded. One more possibility is opened by extending the network design space to the topology as well: instead of changing only the Boolean function characterizing each node, randomly add, change or delete neighbours from nodes. This practice also allows to explore networks with fractional K, where each node may have a different number of

predecessors.

An even more interesting research would be extending the proposed algorithm model to work online, that is, placing the robot in a continuous development context where its internal structure changes continuously until some target requirement, possibly correlated to the deployment environment such that as the environment changes the development restarts, are met. Indeed the challenge is far from easy since the underlying strategy exploited until now is a stochastic descent search, therefore *rapid convergence* may be hard or not always achievable.

Finally, regarding other possible interesting continuations on the tracks laid down by this work, the development of an already whole BN-robot controller that satisfies both `SBN` and `BBN` requirement is surely among those, together with a deployment of the proposed robot controller on a real world context.

Appendices

Appendix A

Selective Boolean Network Constraints

In this appendix are collected all the figures regarding the exploration of the solution space for the Selective Boolean Networks development, as explained in Section 3.2.1. In order to display them correctly and to ease the view experience, the figures are proposed on a whole page scale, with an orientation from inside toward outside.

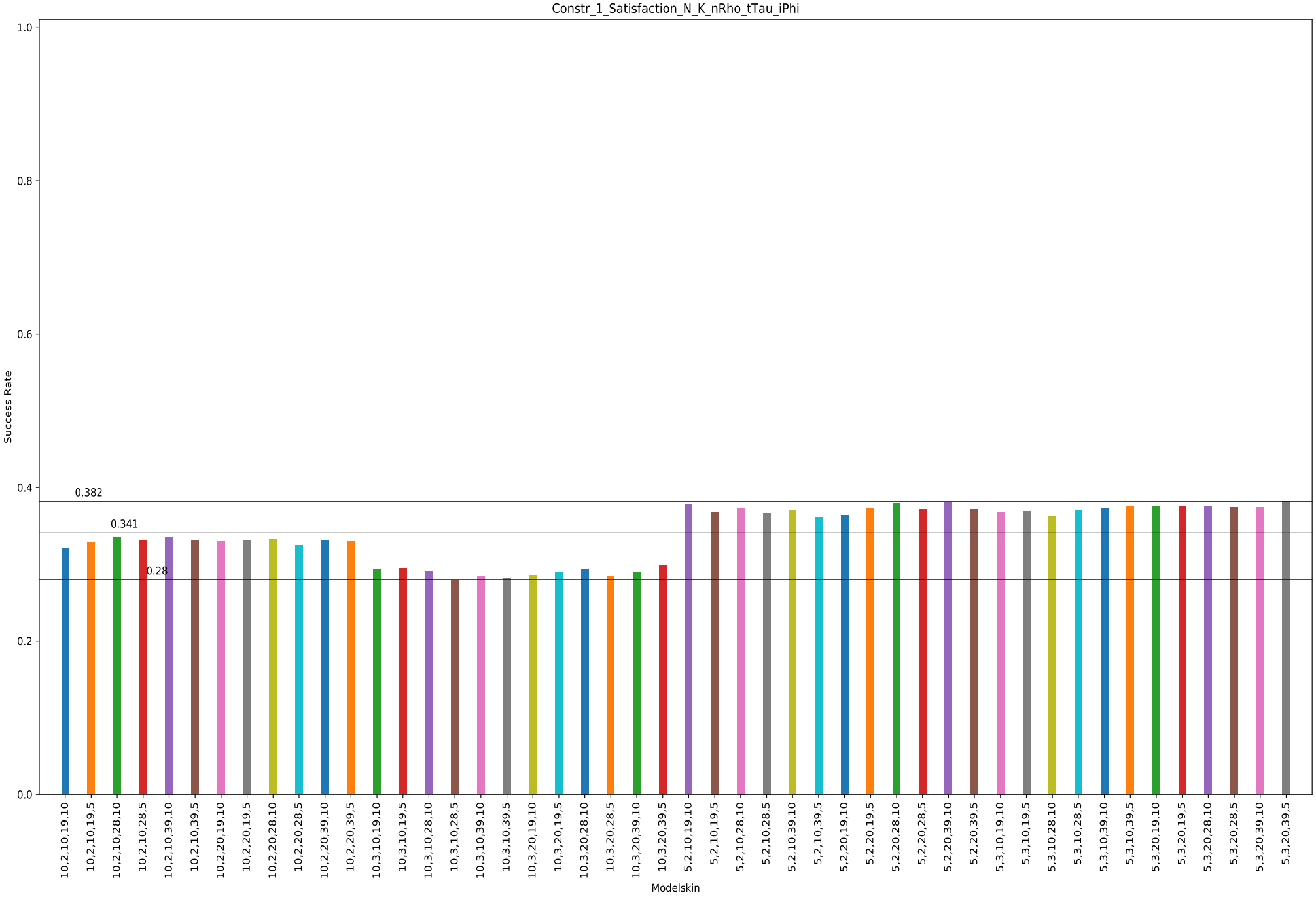

Figure A.1: `Constraint #1` satisfaction on 480000 `RBNs`.

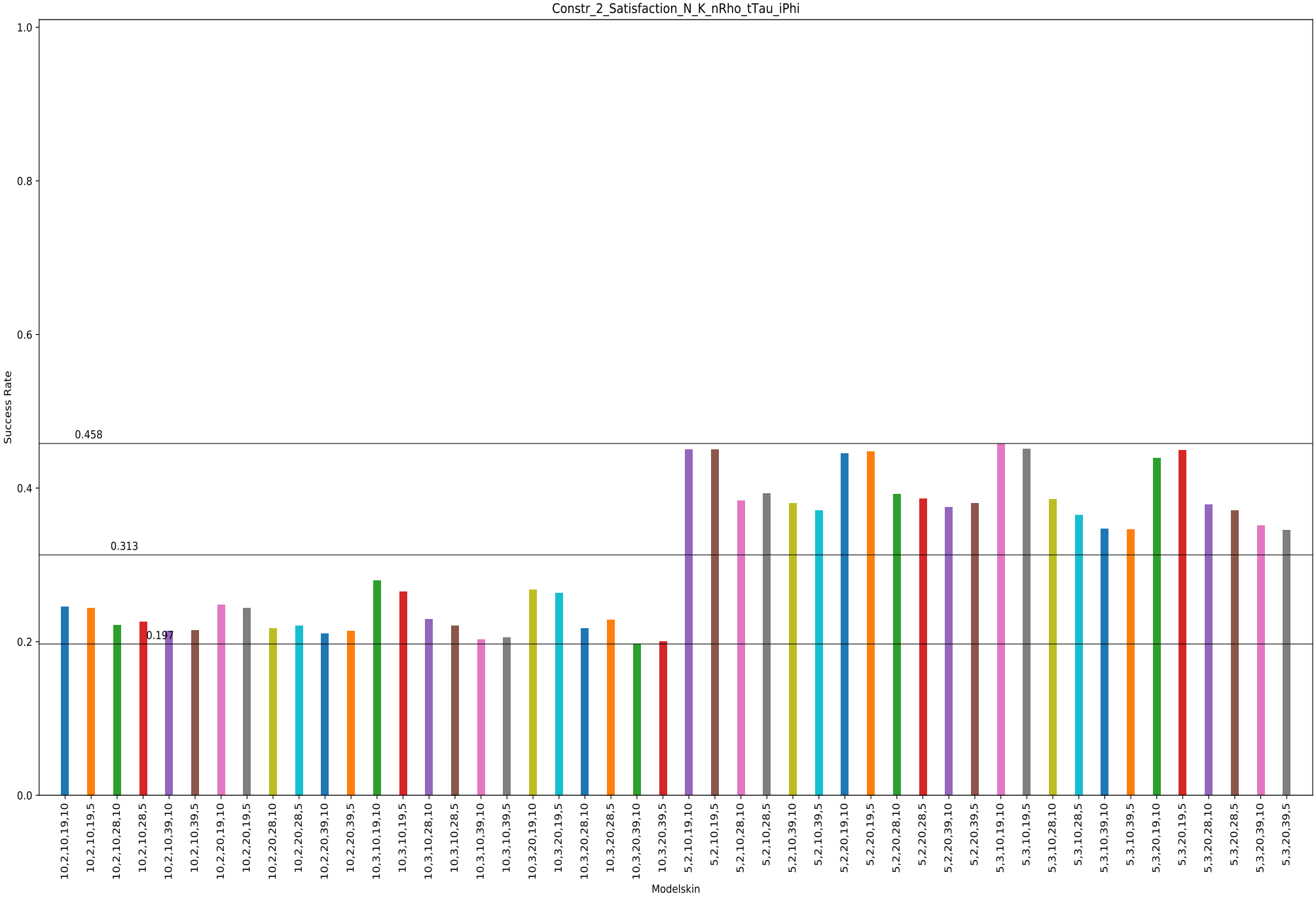

Figure A.2: `Constraint #2` satisfaction on 480000 `RBNs`.

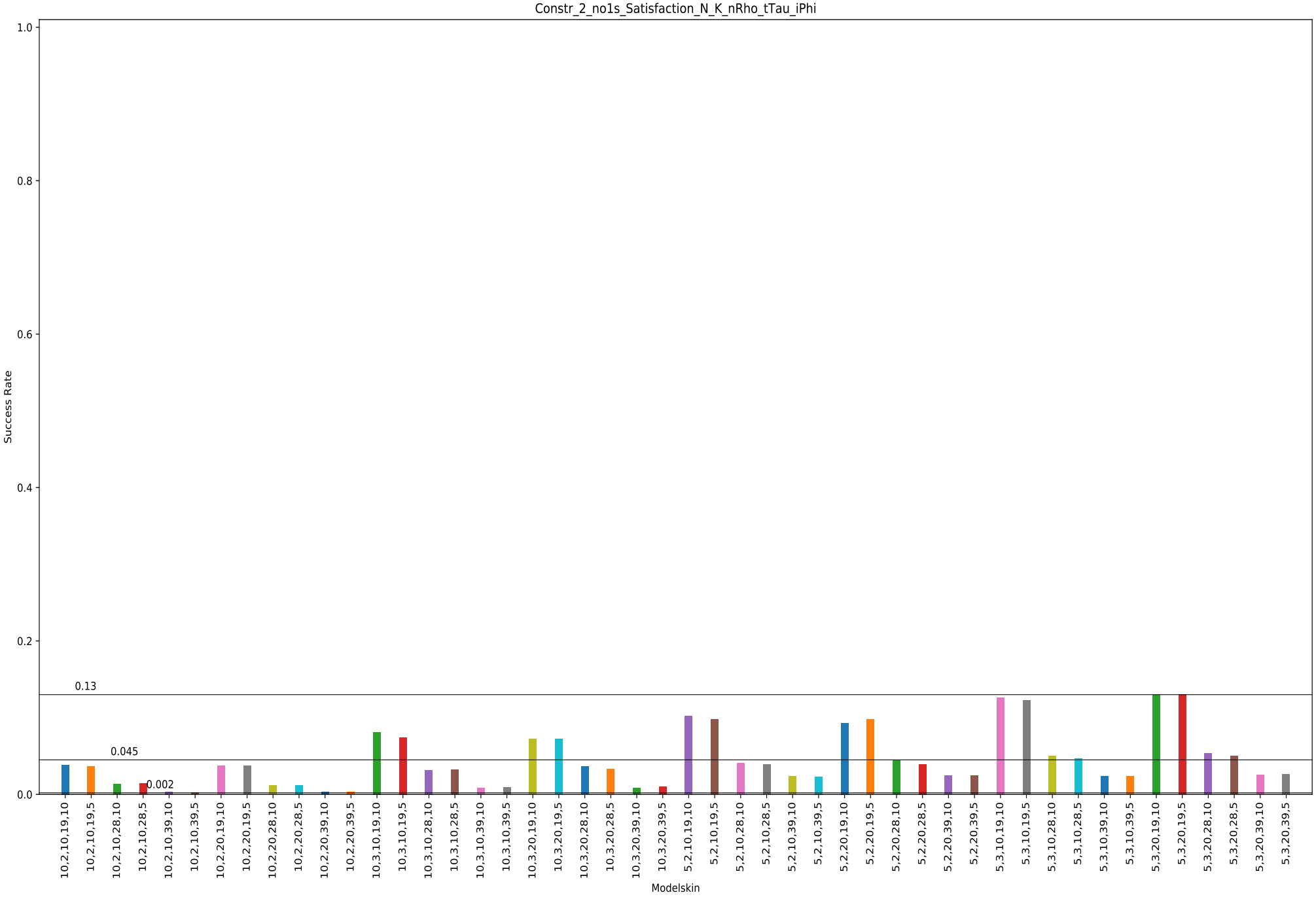

Figure A.3: `Constraint #2` with no network with a single attractor satisfaction on 480000 `RBNs`.

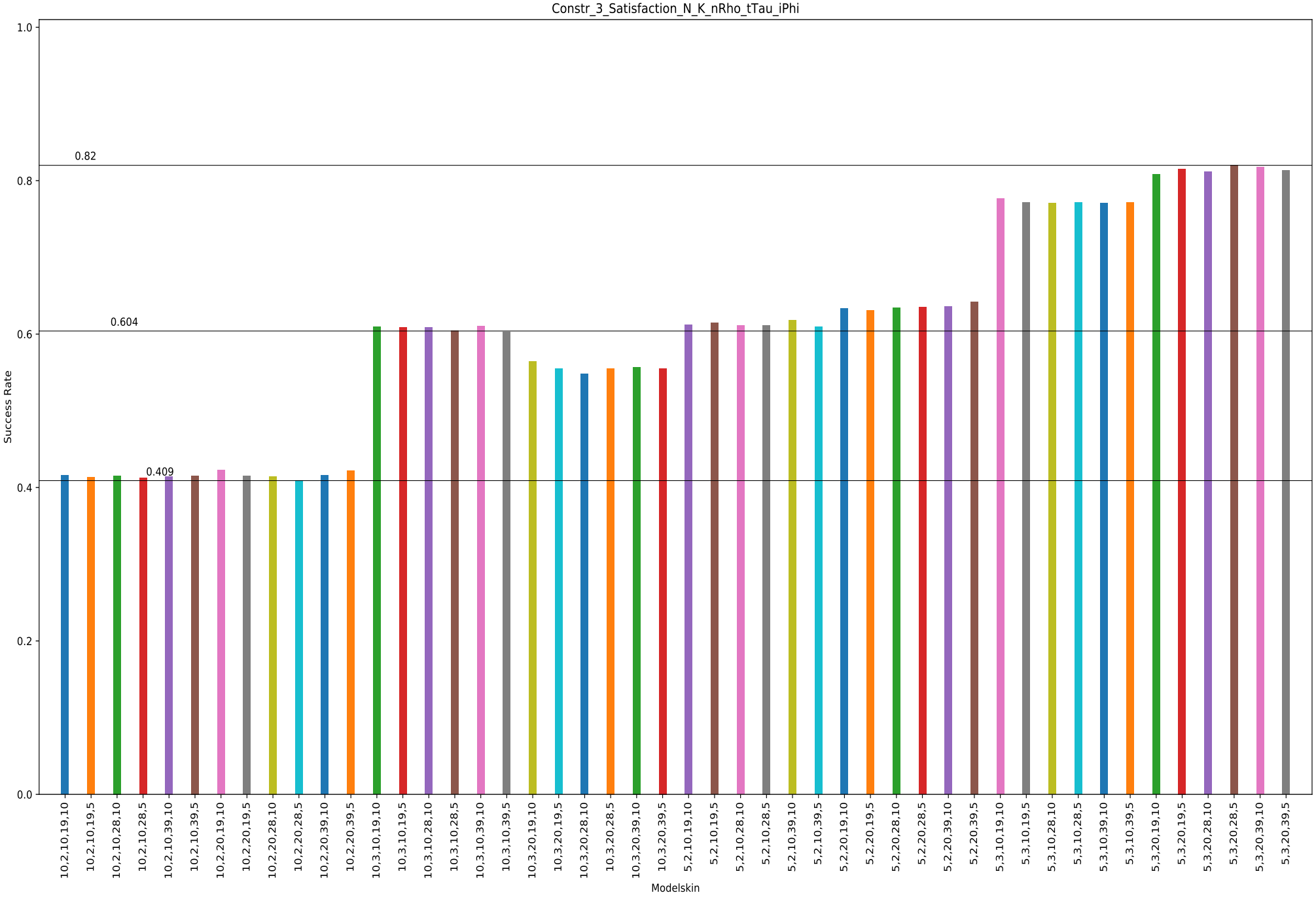

Figure A.4: `Constraint #3` satisfaction on 480000 `RBNs`.

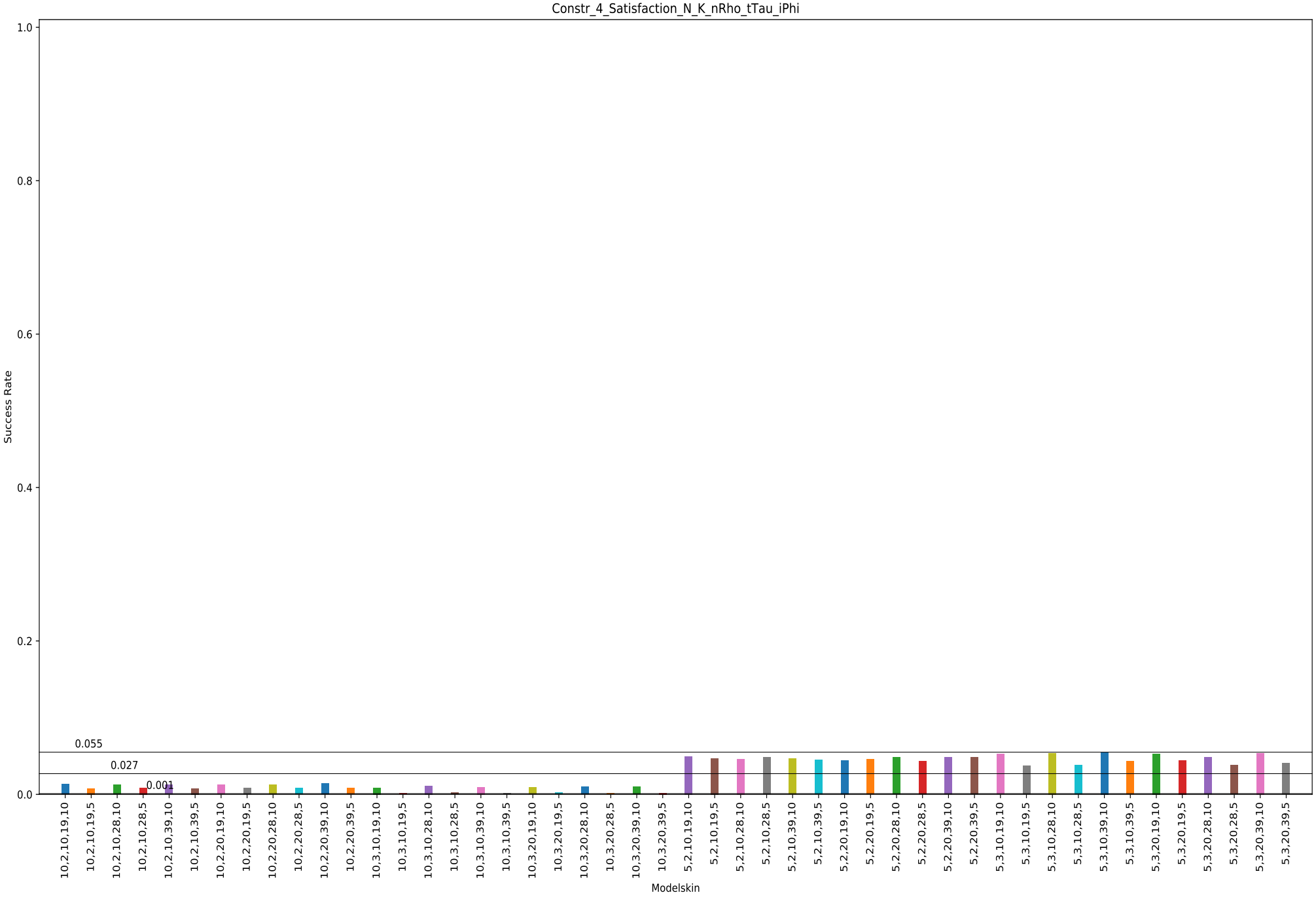

Figure A.5: `Constraint #4` satisfaction on 480000 `RBNs`.

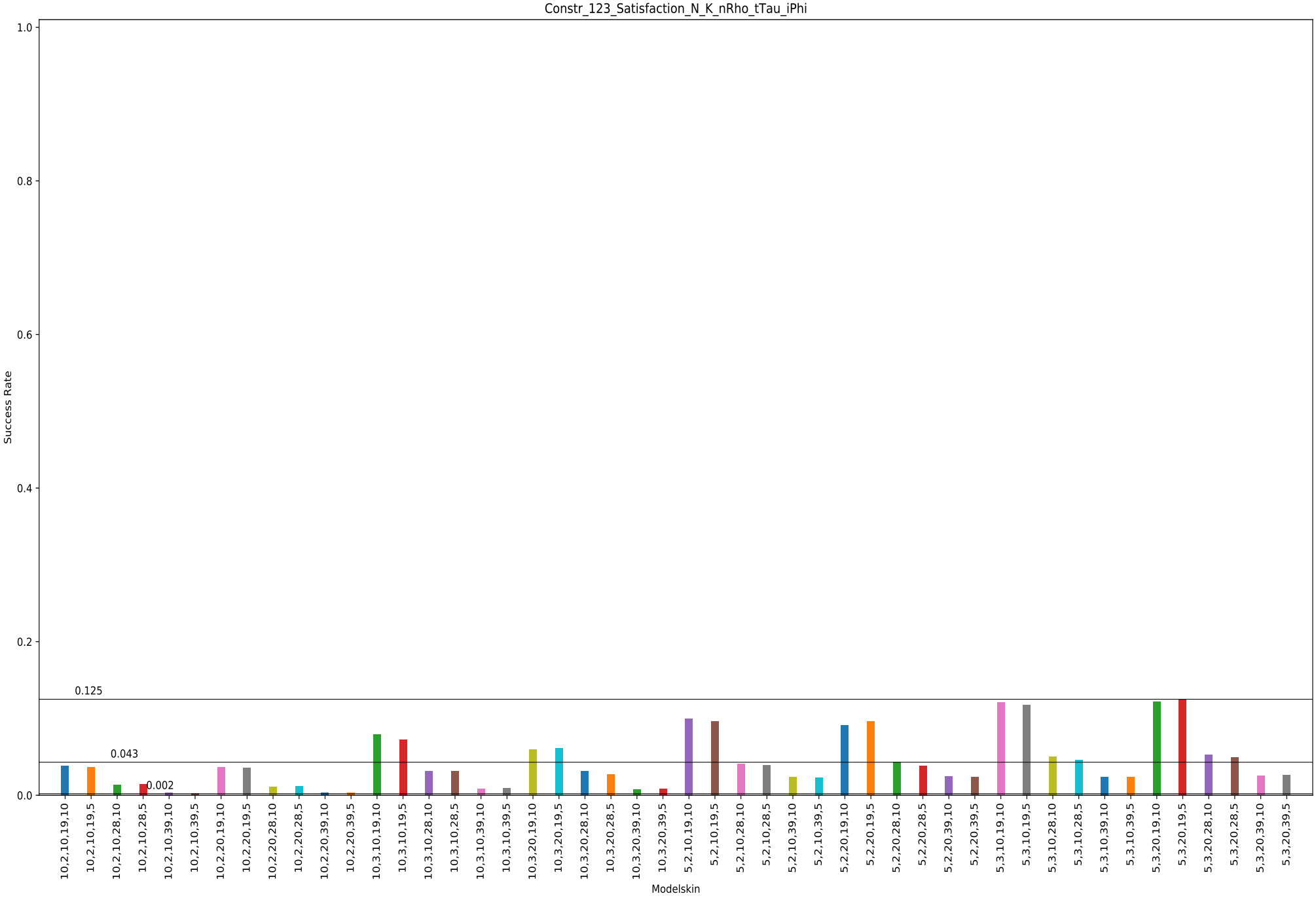

Figure A.6: `Constraint #1`, `#2` and `#3` satisfaction on 480000 `RBNs`.

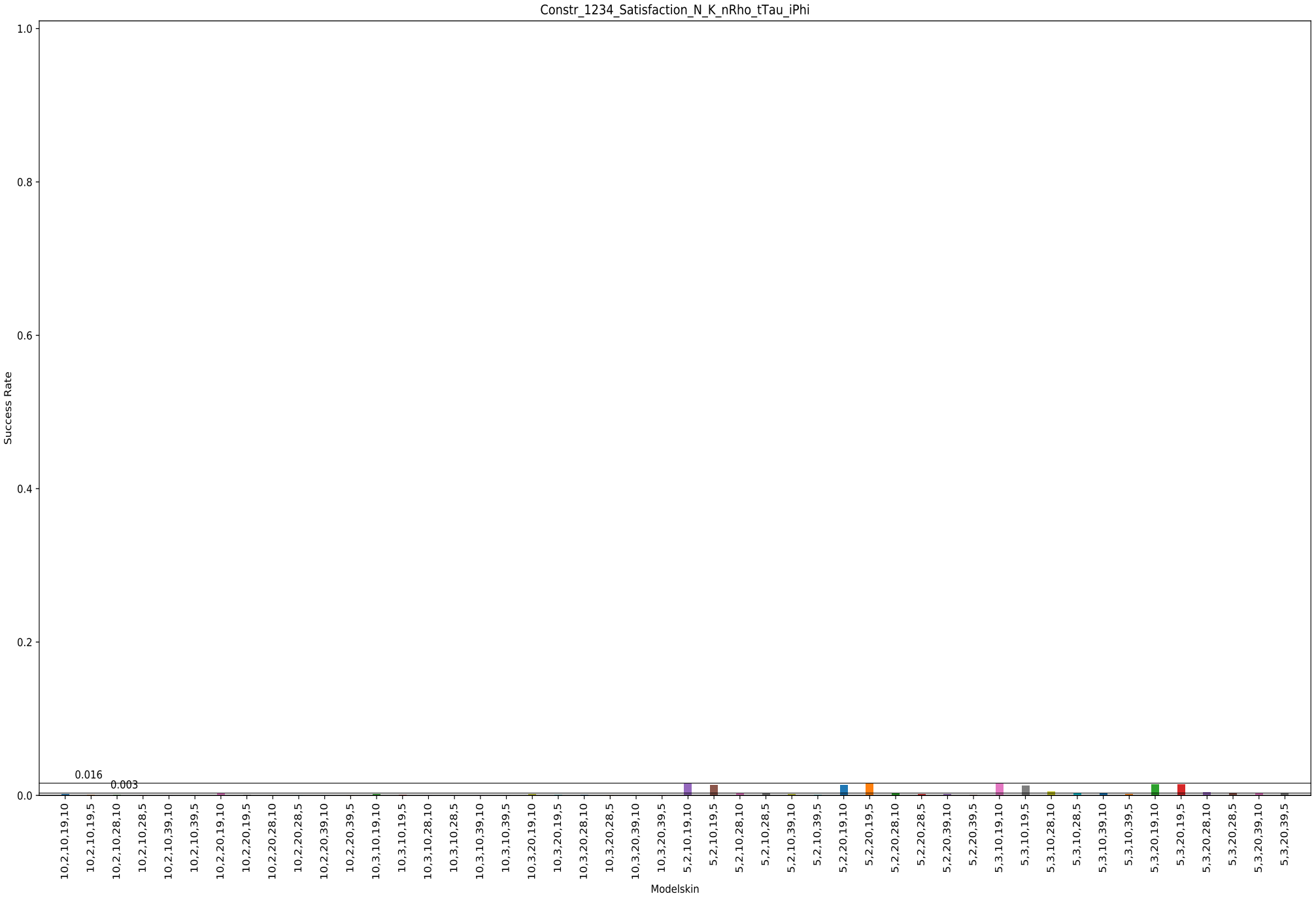

Figure A.7: All Constraints satisfaction on 480000 RBNs.

Appendix B

Boolean Network Controller: Phototaxis

In this appendix are collected all the figures regarding the development of `Behavioural Boolean Networks` expressing `phototaxis`, as explained in Section 3.2.2. In order to display them correctly and to ease the view experience, the figures are proposed on a whole page scale, with an orientation from inside toward outside.

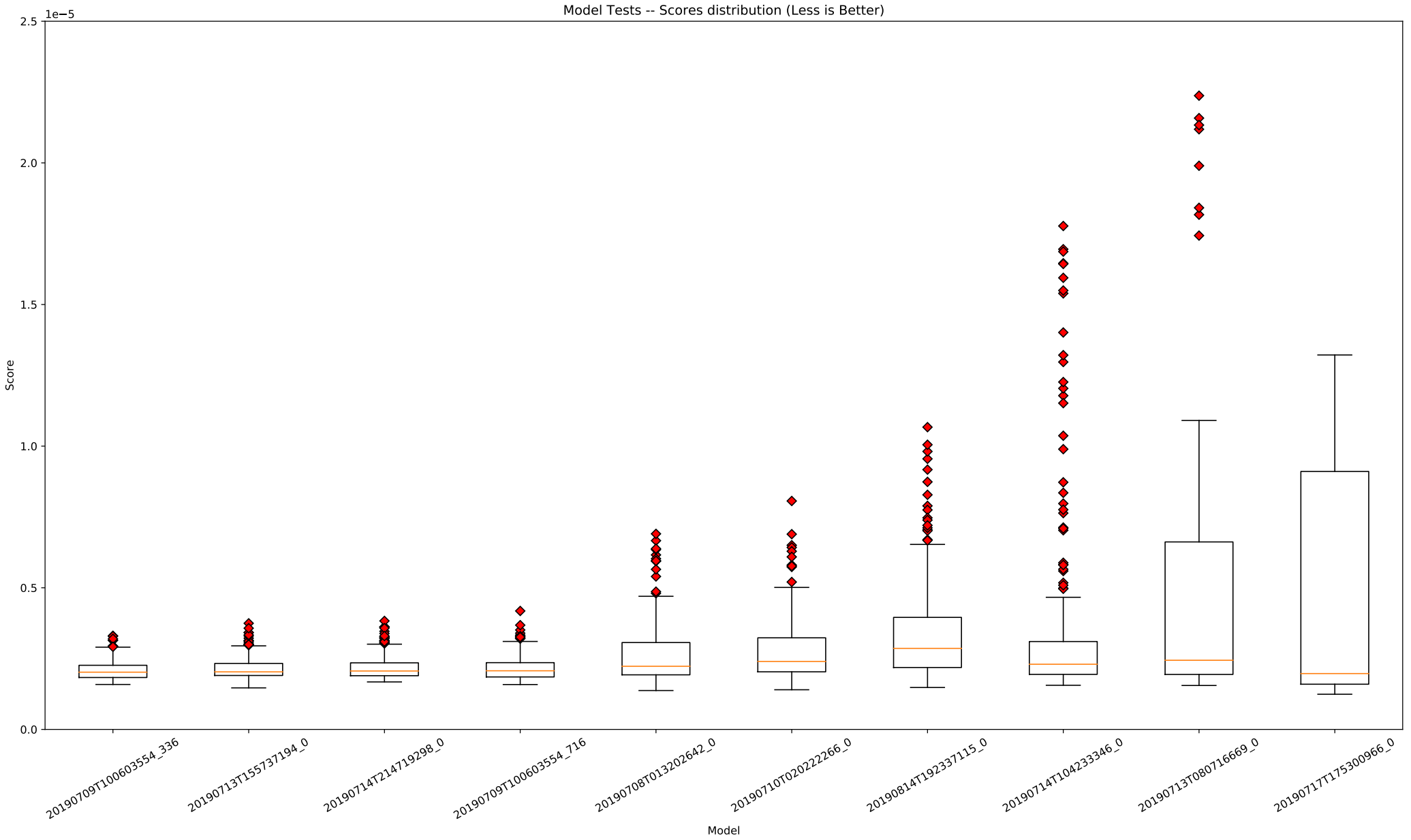

Figure B.1: Box-plot: each networks `phototaxis` scores achieved during the test phase.

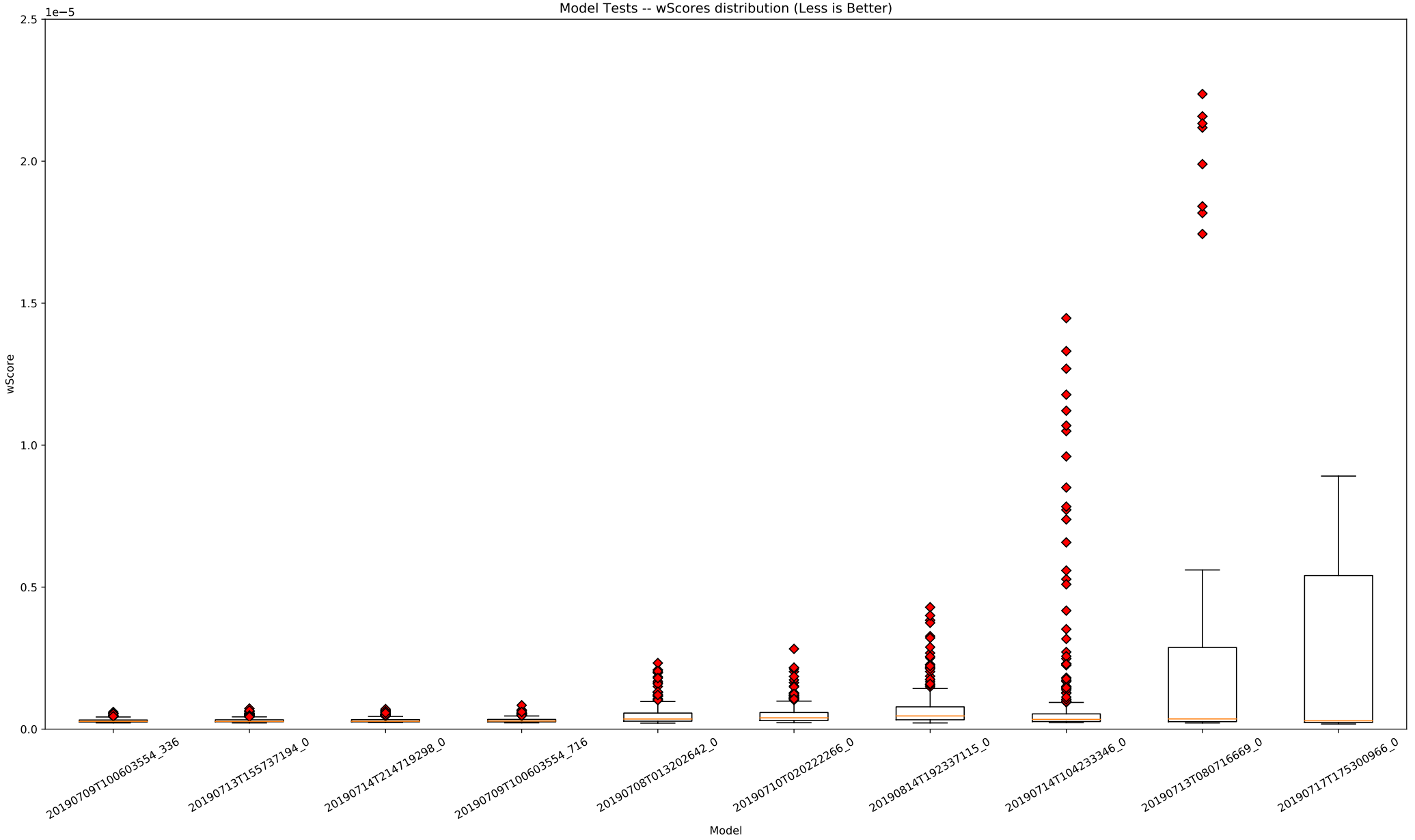

Figure B.2: Box-plot: each network weighted `phototaxis` scores achieved during the test phase.

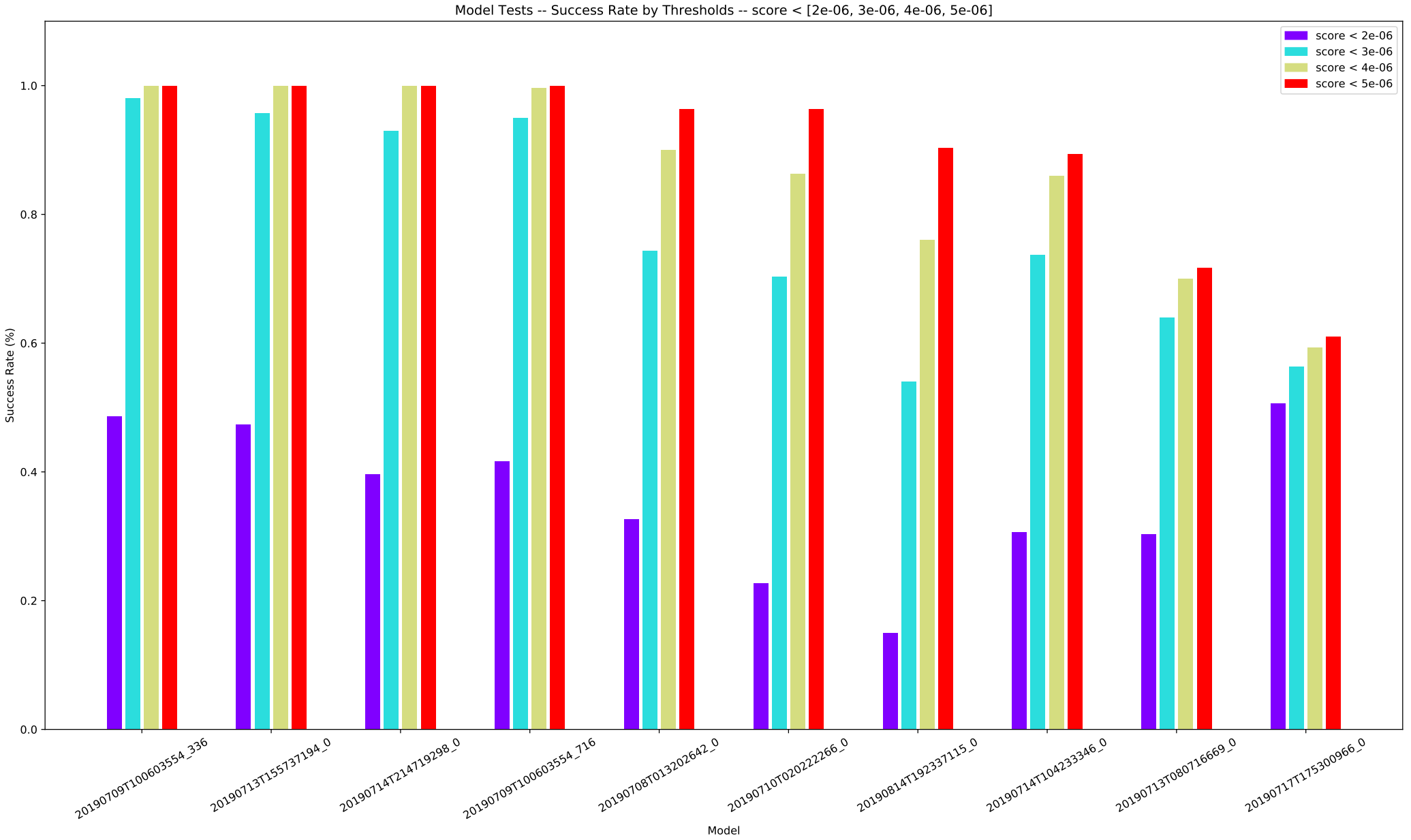

Figure B.3: Bar-plot: each network `phototaxis` scores is tested against a set of optimal scores.

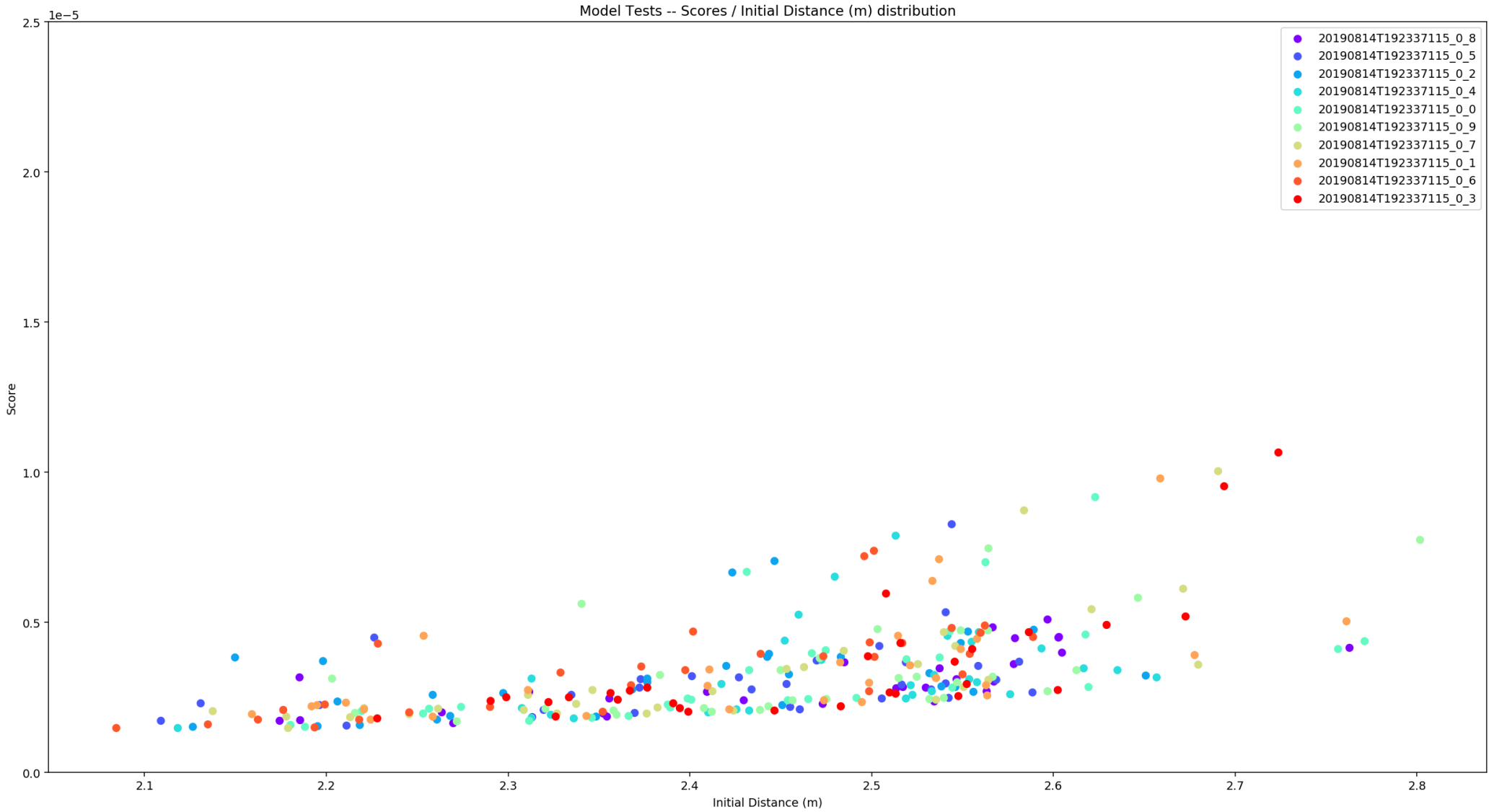

Figure B.4: Scatter-plot: network `20190814T192337115`, , x-axis → initial distance, y-axis → `phototaxis` scores

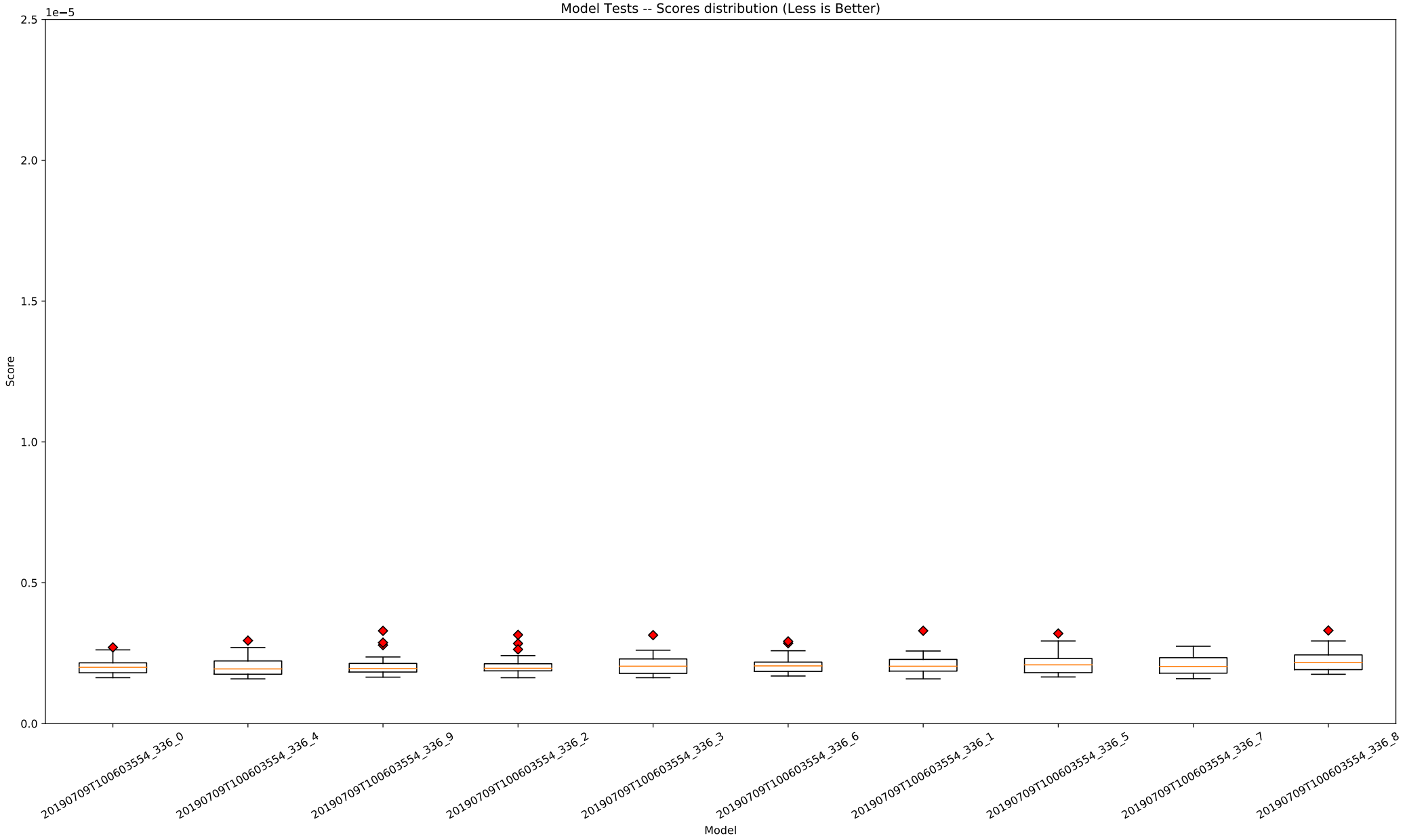

Figure B.5: Box-plot: network 20190709T100603554_336 `phototaxis` test scores.

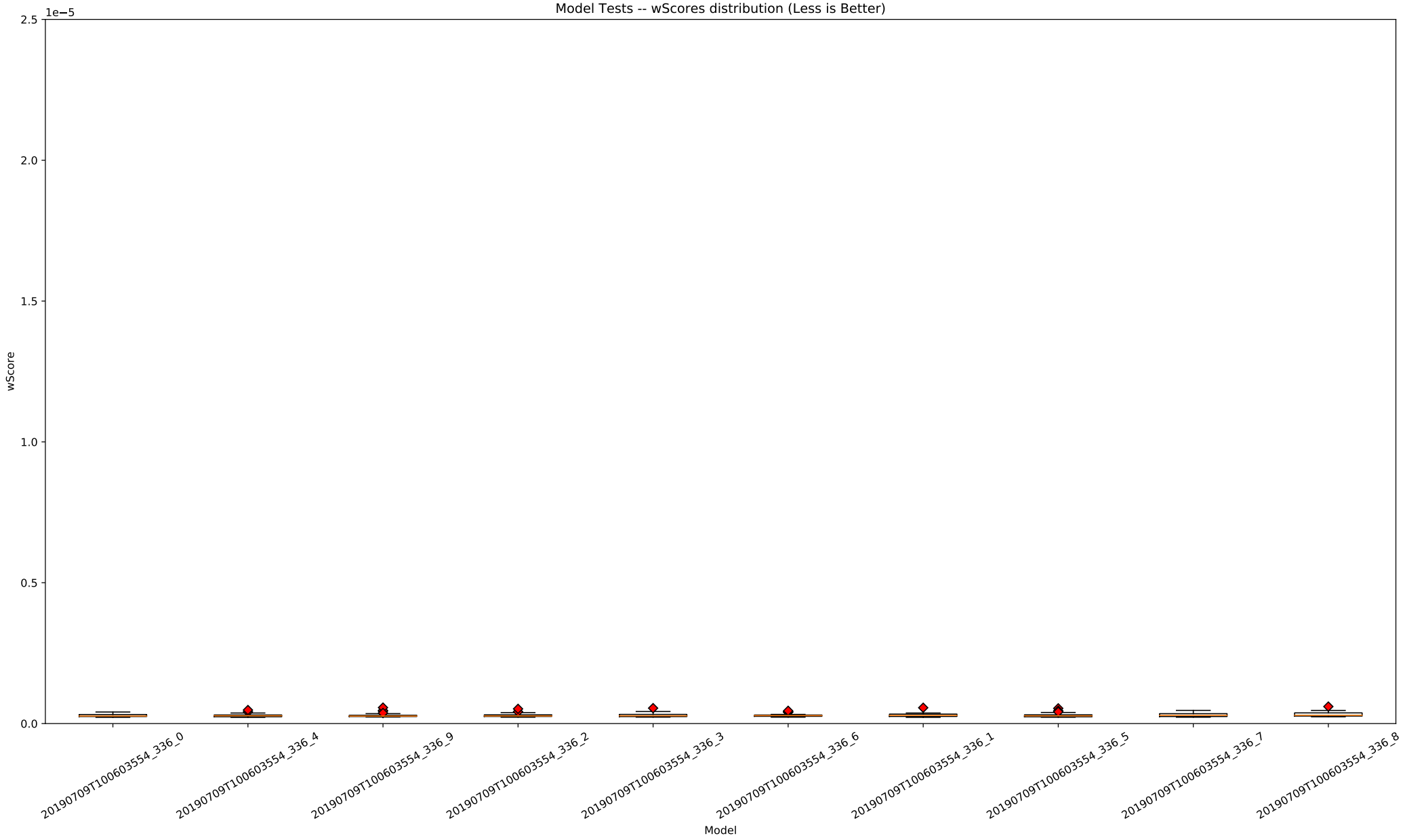

Figure B.6: Box-plot: network 20190709T100603554_336 weighted `phototaxis` test scores.

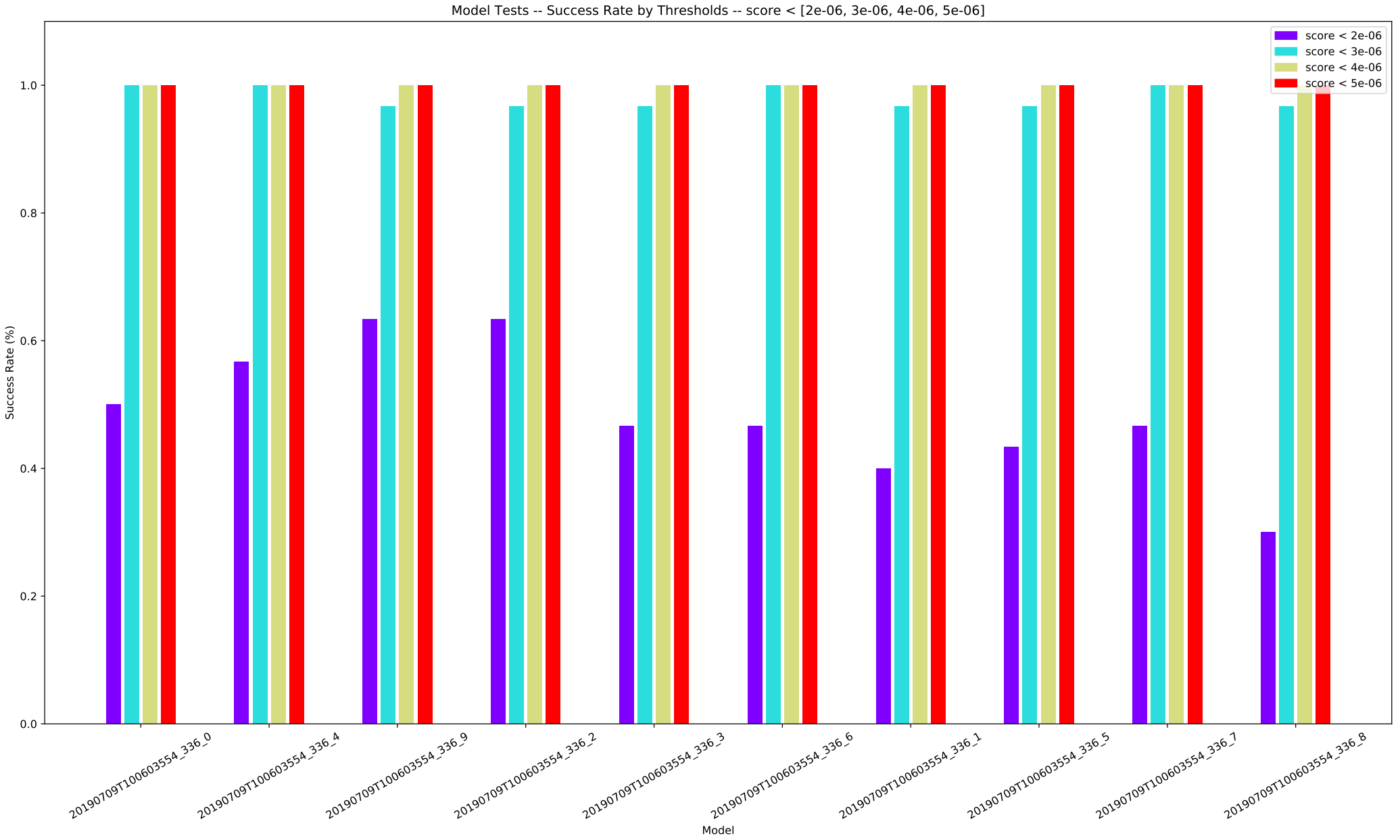

Figure B.7: Bar-plot: network `20190709T100603554_336`, each instance `phototaxis` scores are tested against a set of optimal scores.

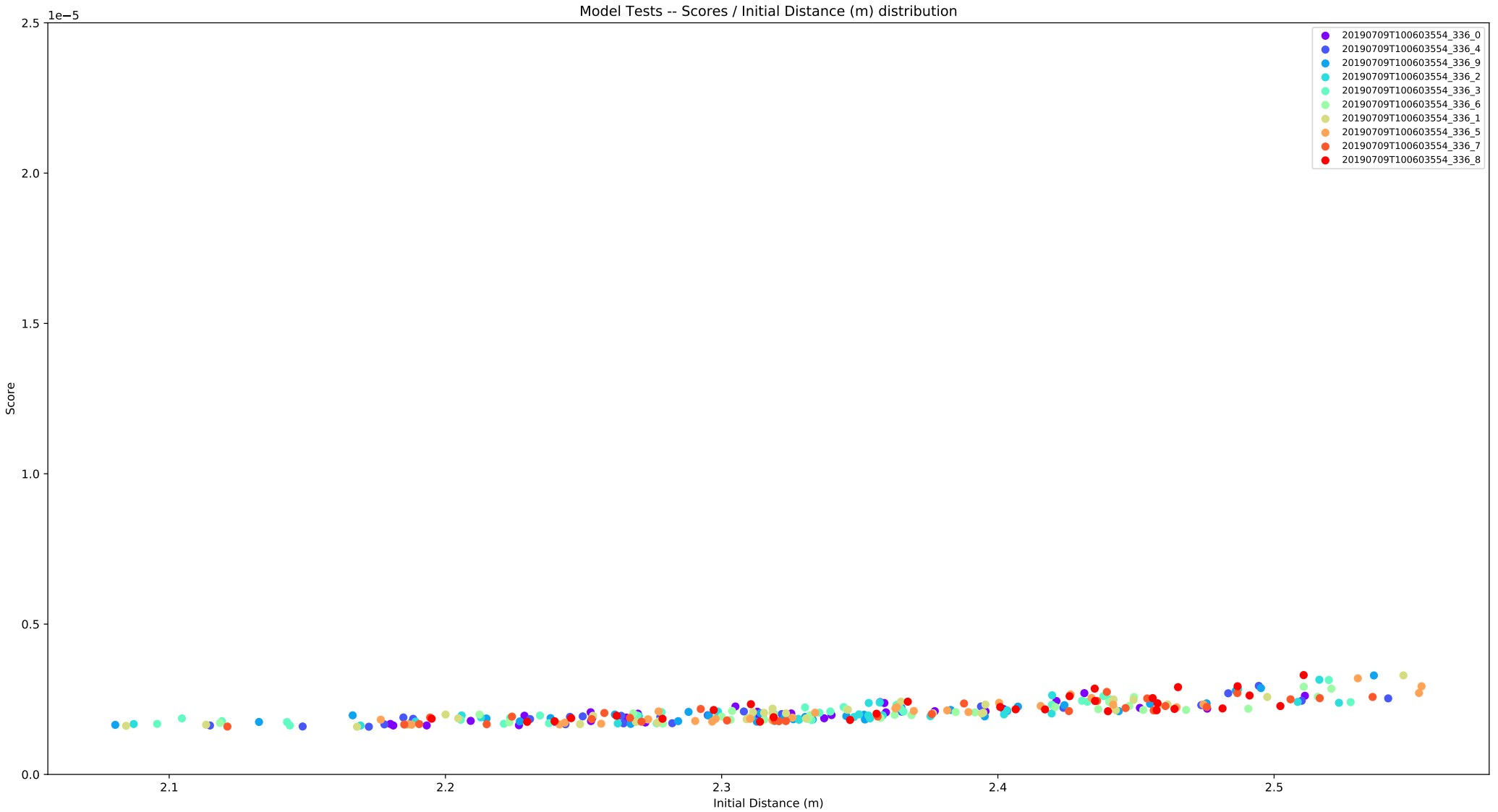

Figure B.8: Scatter-plot: network 20190709T100603554_336, x-axis → initial distance, y-axis → `phototaxis` scores

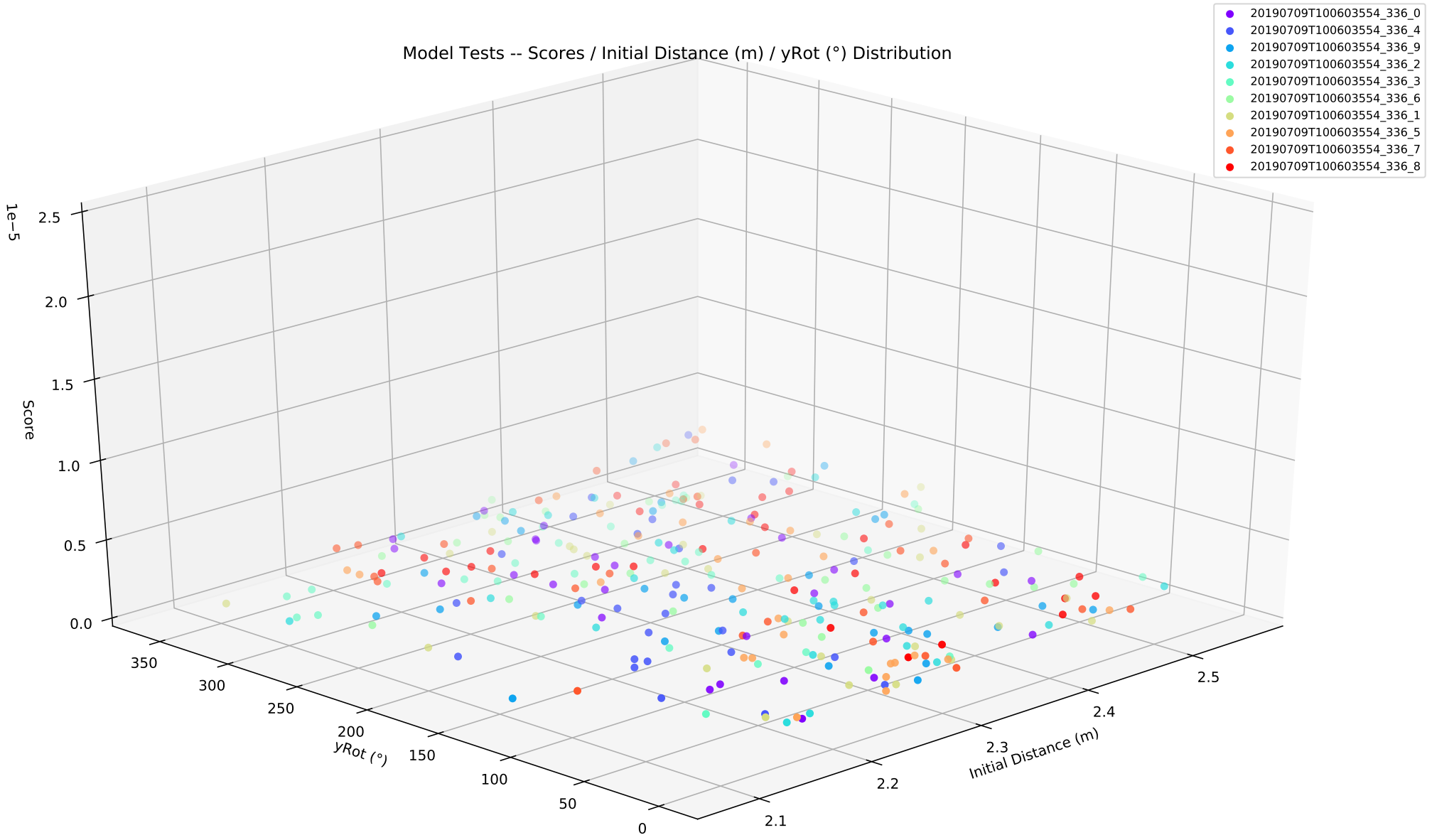

Figure B.9: Scatter-plot: network `20190709T100603554_336`, x-axis → initial distance, y-axis → agent rotation, z-axis → `phototaxis` scores

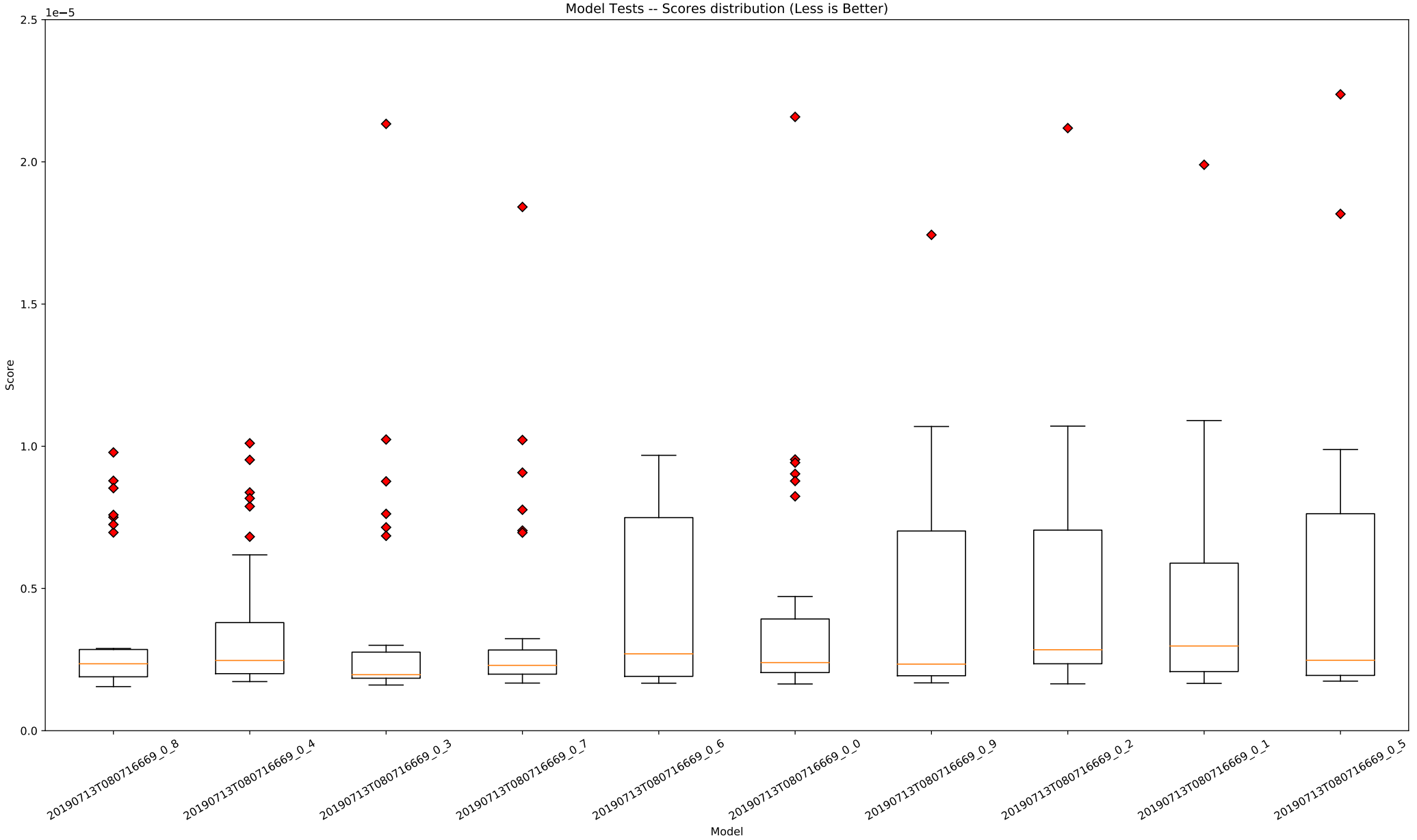

Figure B.10: Box-plot: network `20190713T080716669` `phototaxis` test scores.

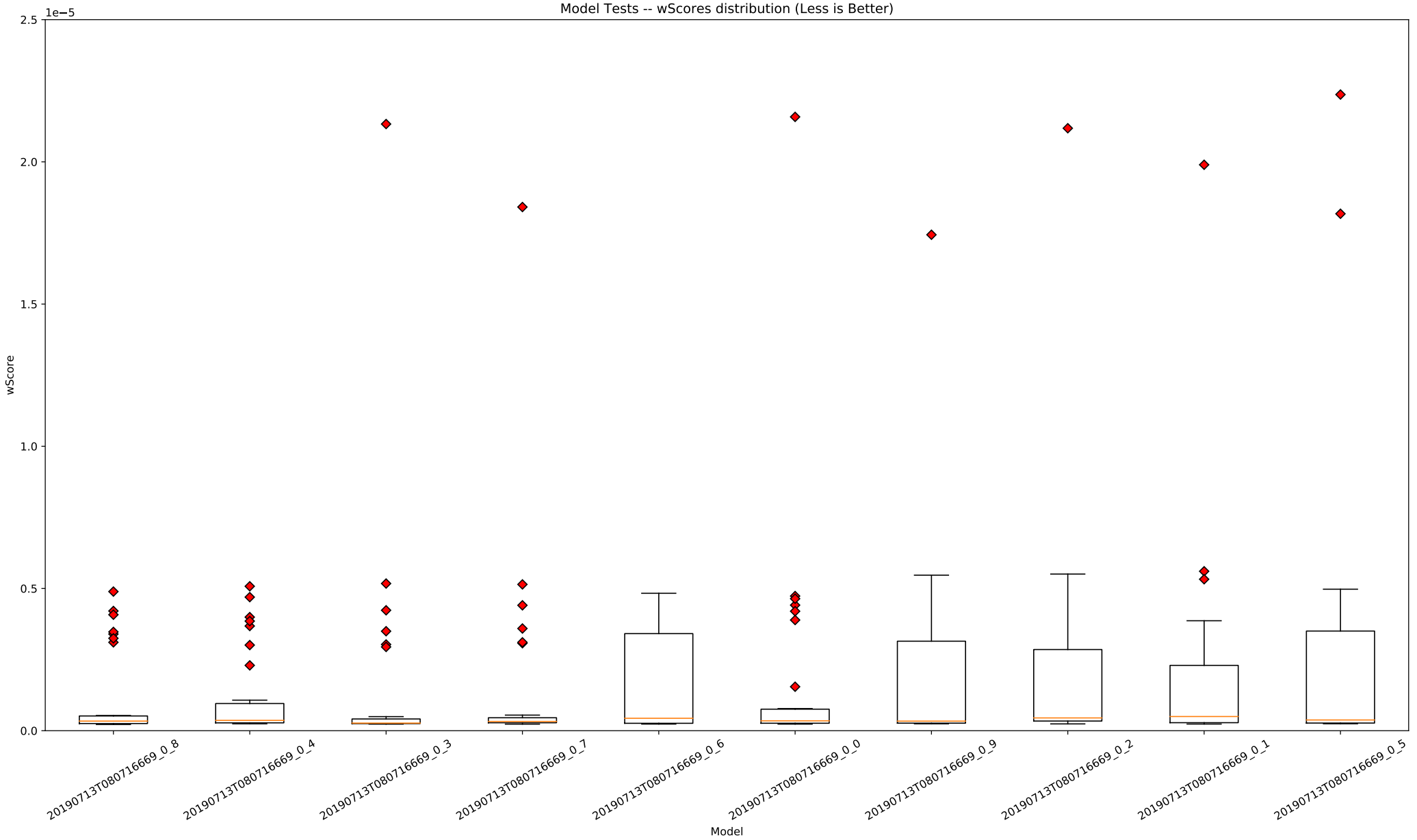

Figure B.11: Box-plot: network `20190713T080716669` weighted `phototaxis` test scores.

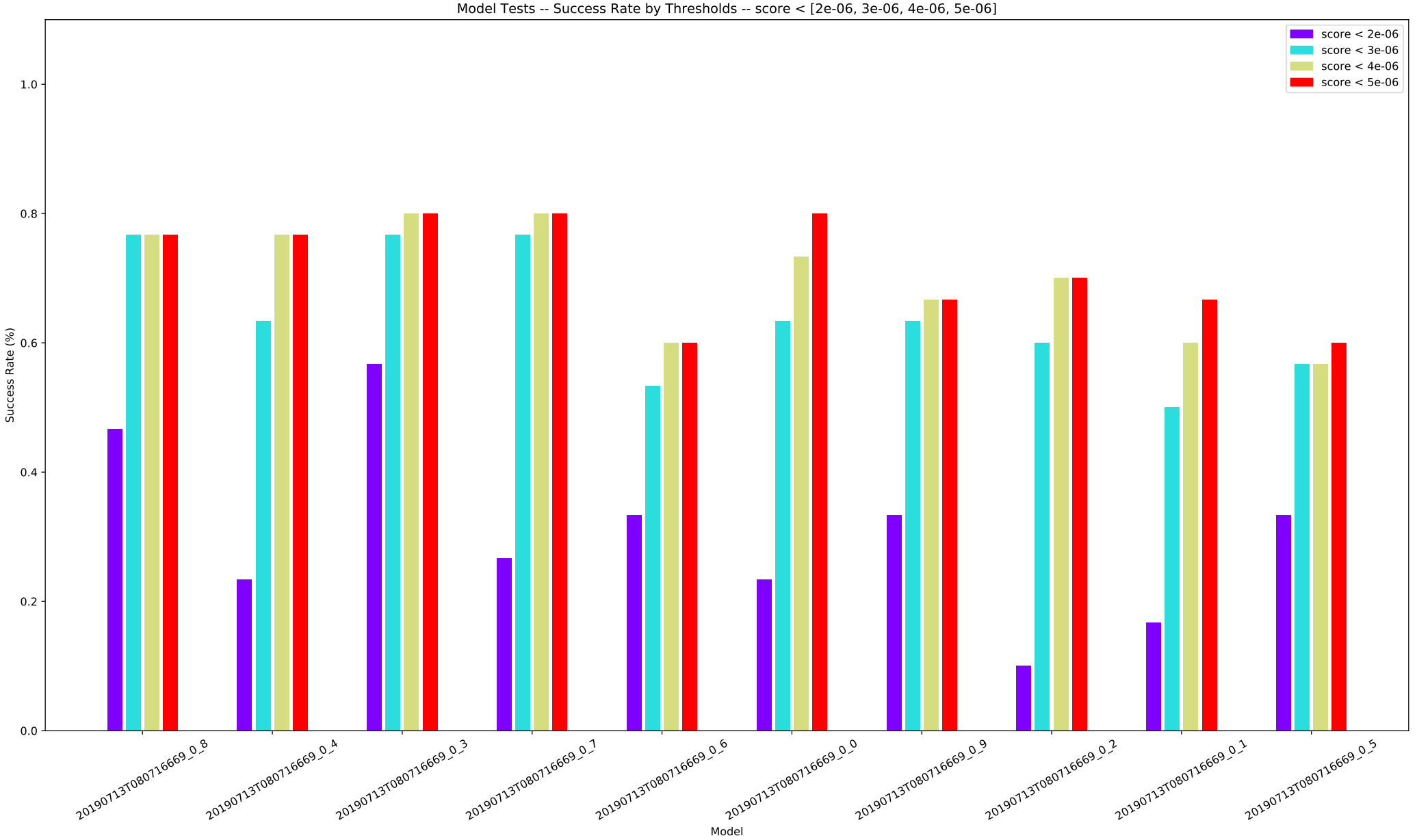

Figure B.12: Bar-plot: network `20190713T080716669` each network `phototaxis` score is tested against a set of optimal scores.

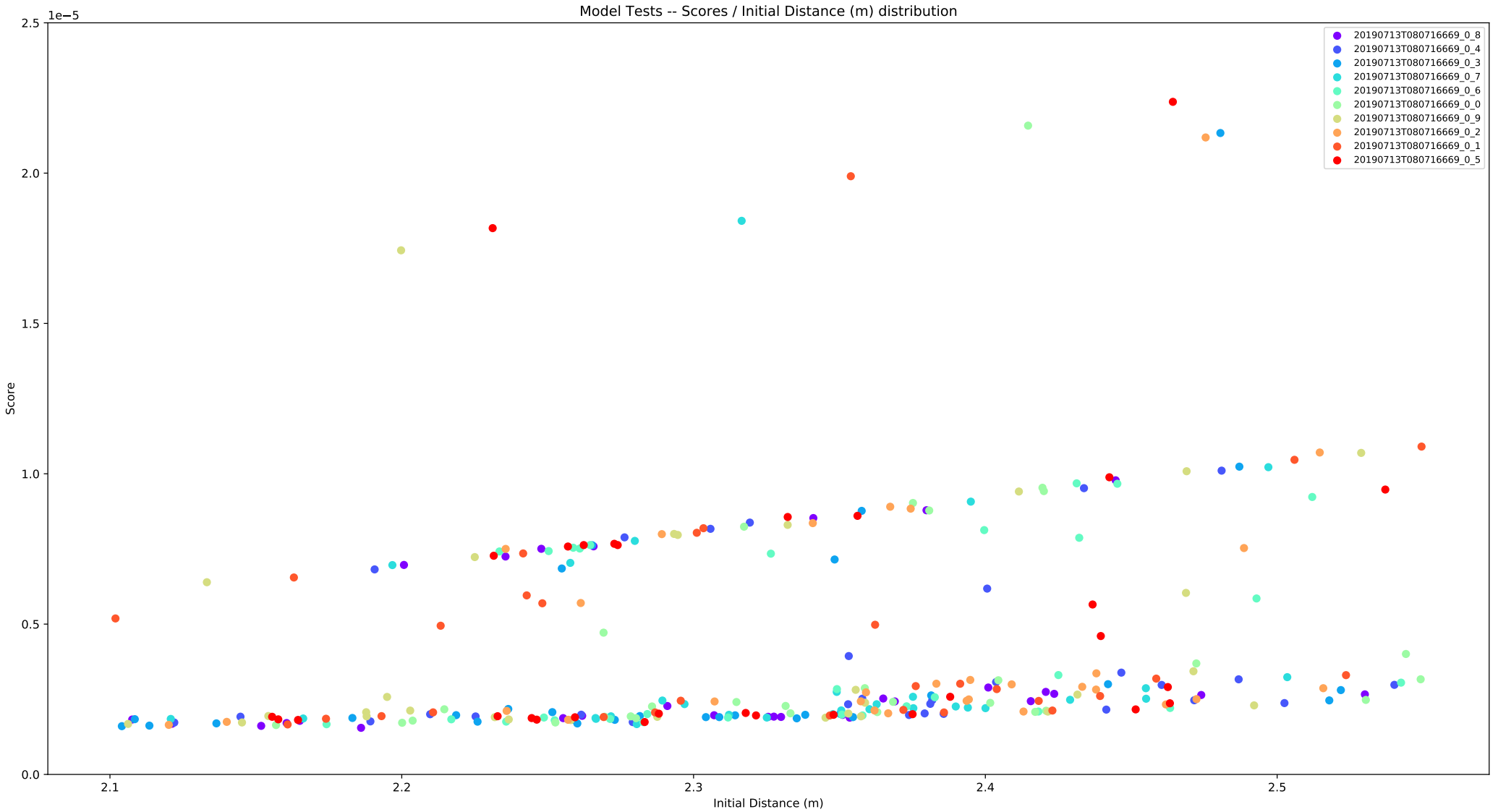

Figure B.13: Scatter-plot: network `20190713T080716669`, x-axis → initial distance, y-axis → `phototaxis` scores

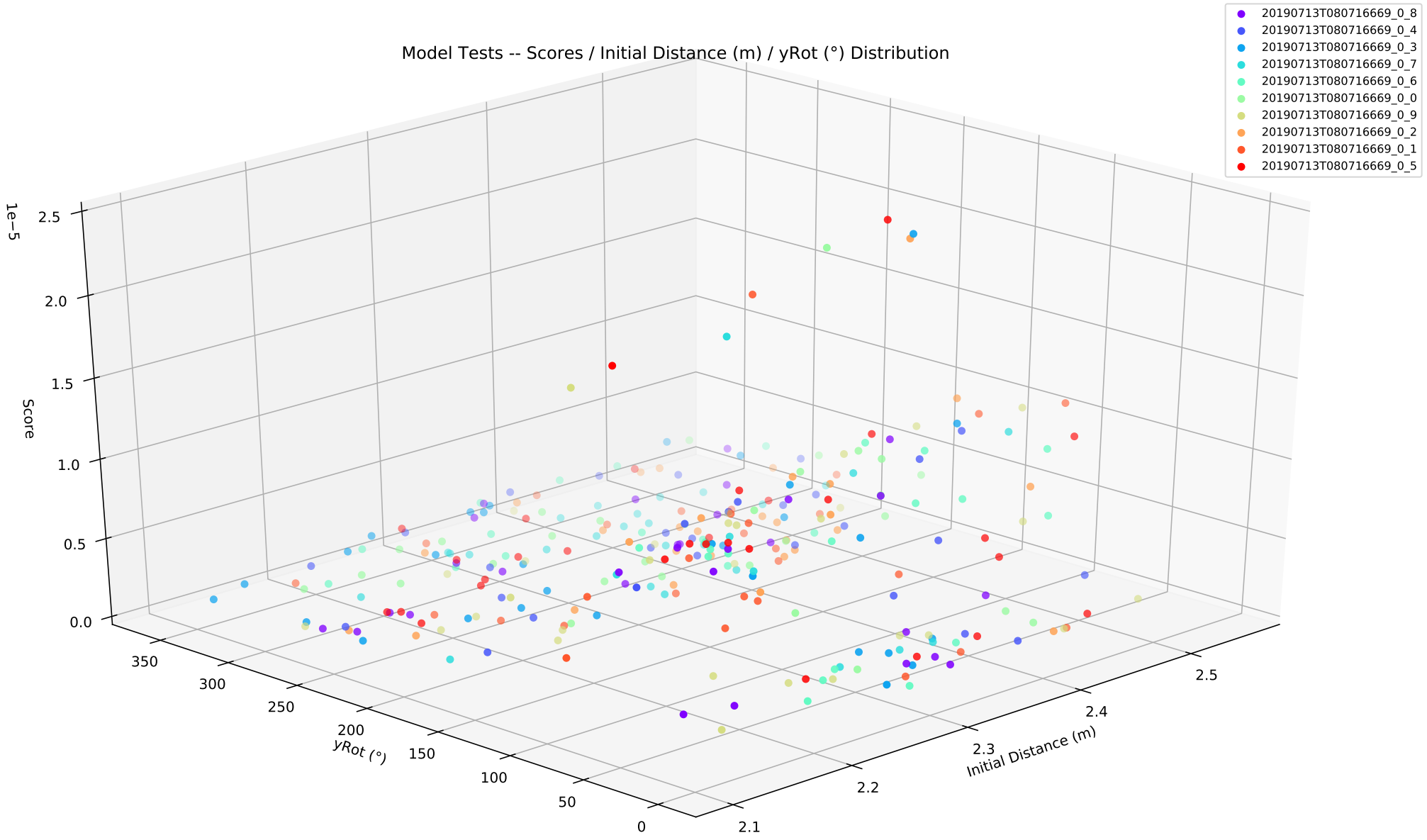

Figure B.14: Scatter-plot: network `20190713T080716669`, x-axis → initial distance, y-axis → agent rotation, z-axis → `phototaxis` scores

Appendix C

Boolean Network Controller: Anti-Phototaxis

In this appendix are collected all the figures regarding the development of `Behavioural Boolean Networks` expressing `anti-phototaxis`, as explained in 3.2.2. In order to display them correctly and to ease the view experience, the figures are proposed on a whole page scale, with an orientation from inside toward outside.

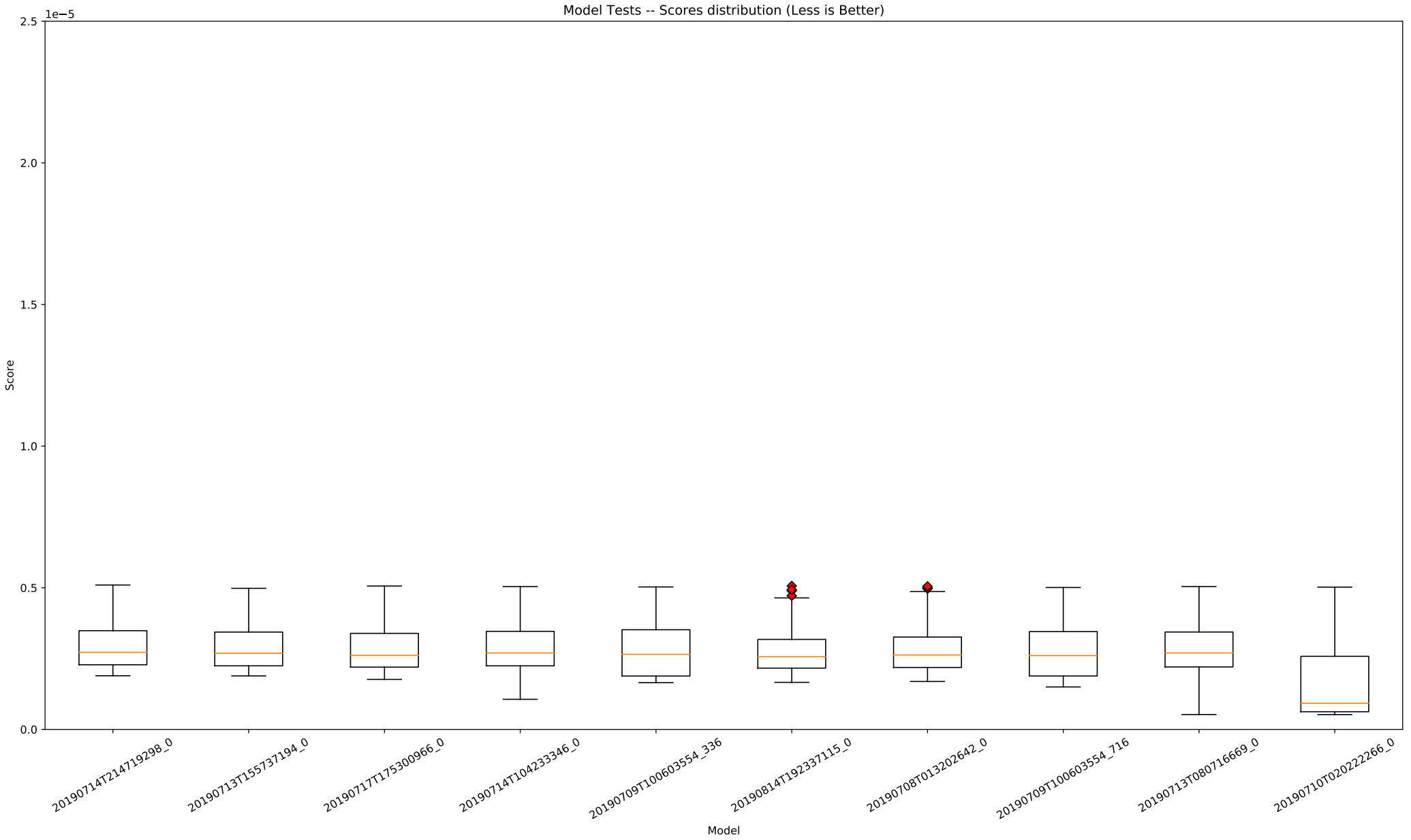

Figure C.1: Box-plot: each network `anti-phototaxis` scores achieved during the test phase.

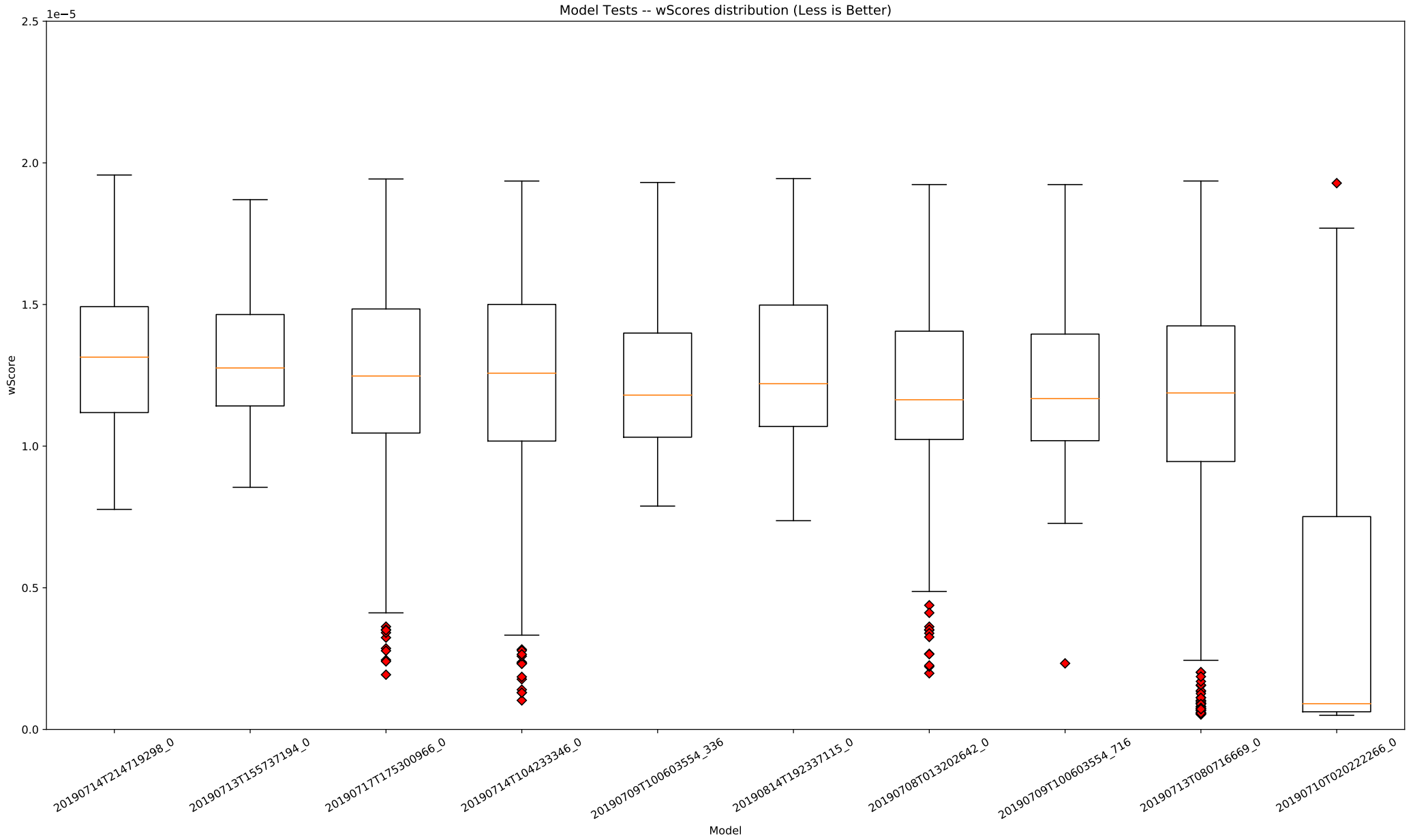

Figure C.2: Box-plot: each network weighted `anti-phototaxis` scores achieved during the test phase.

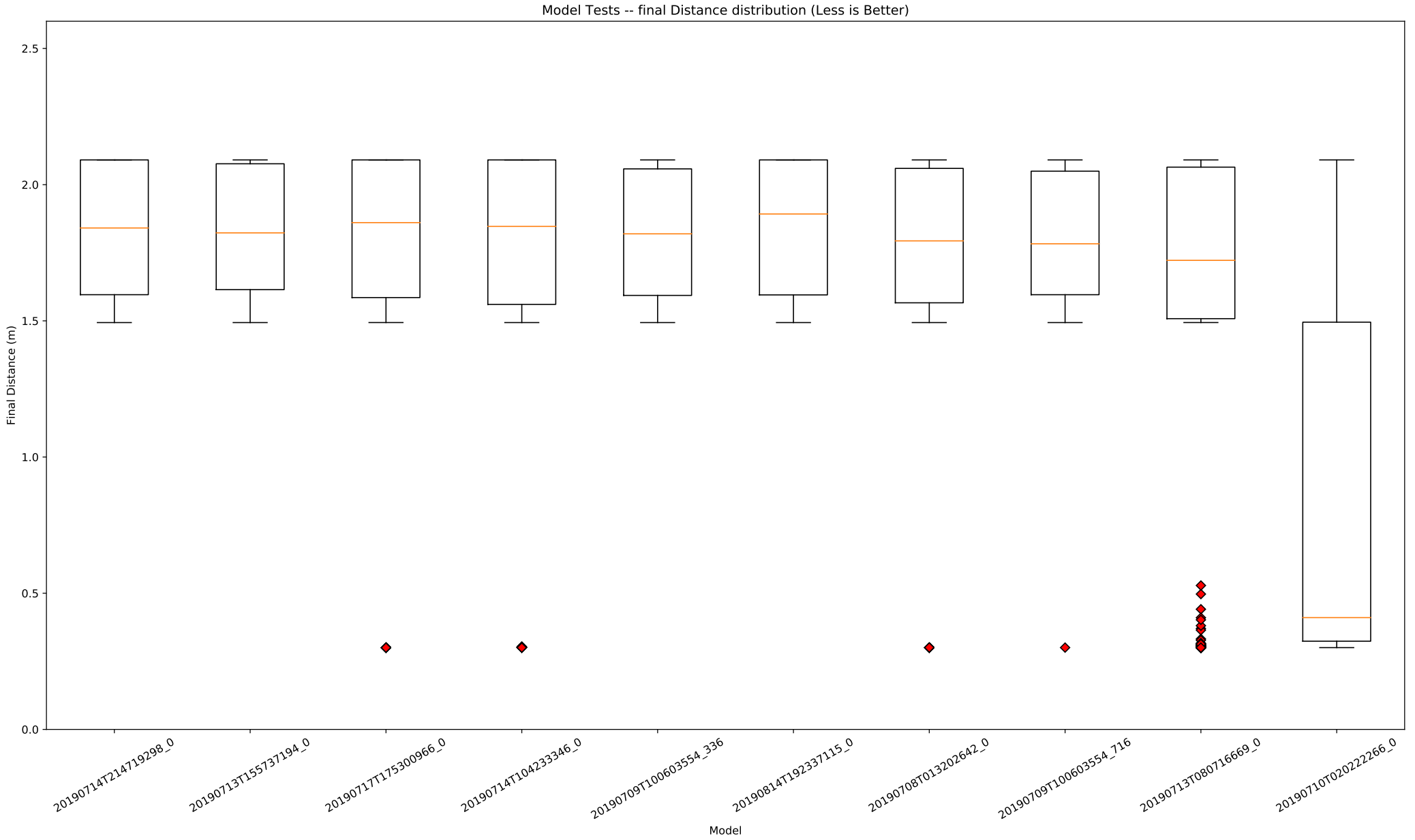

Figure C.3: Box-plot: each network `anti-phototaxis` final distances achieved during the test phase.

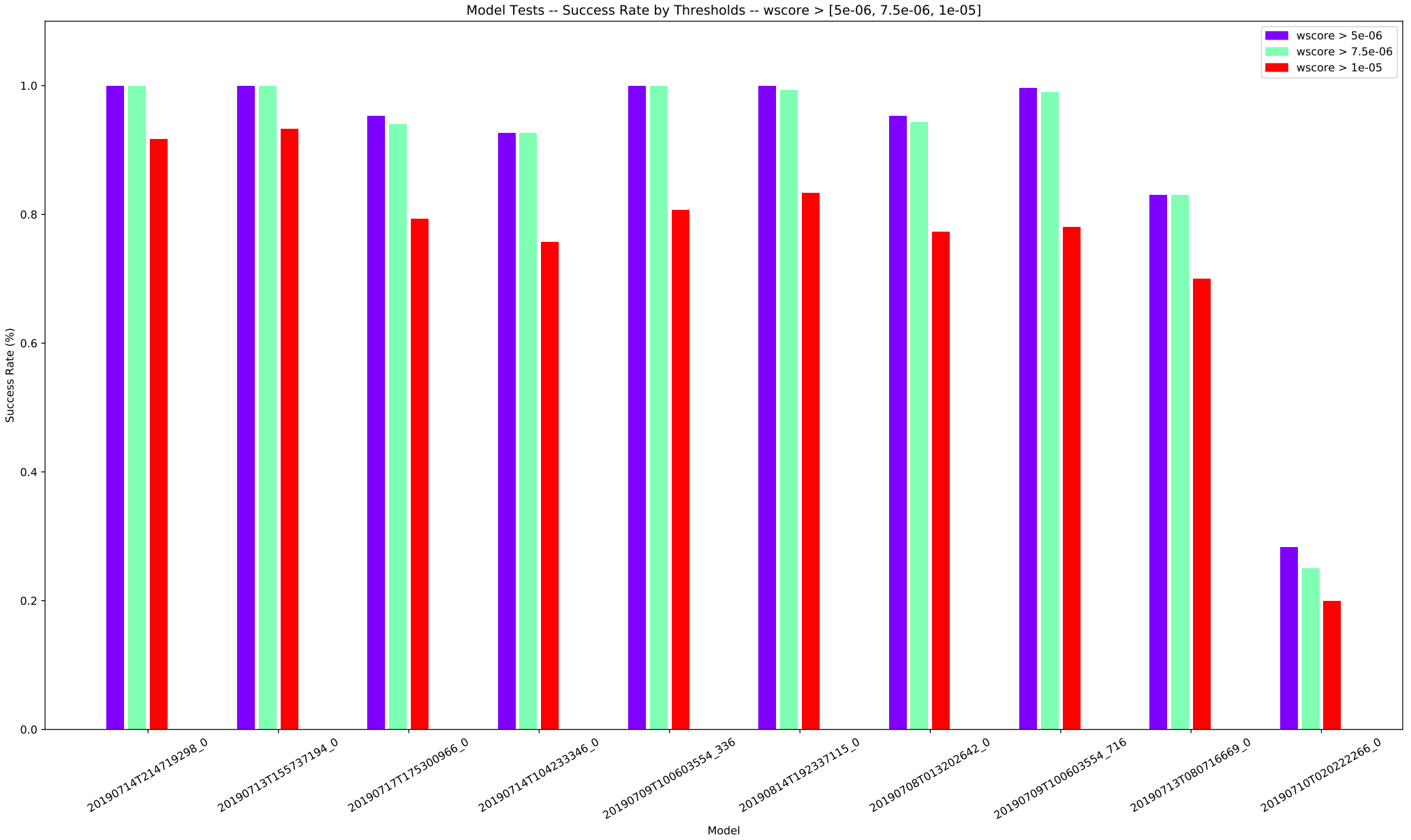

Figure C.4: Bar-plot: each network `anti-phototaxis` weighted scores are tested against a set of optimal scores.

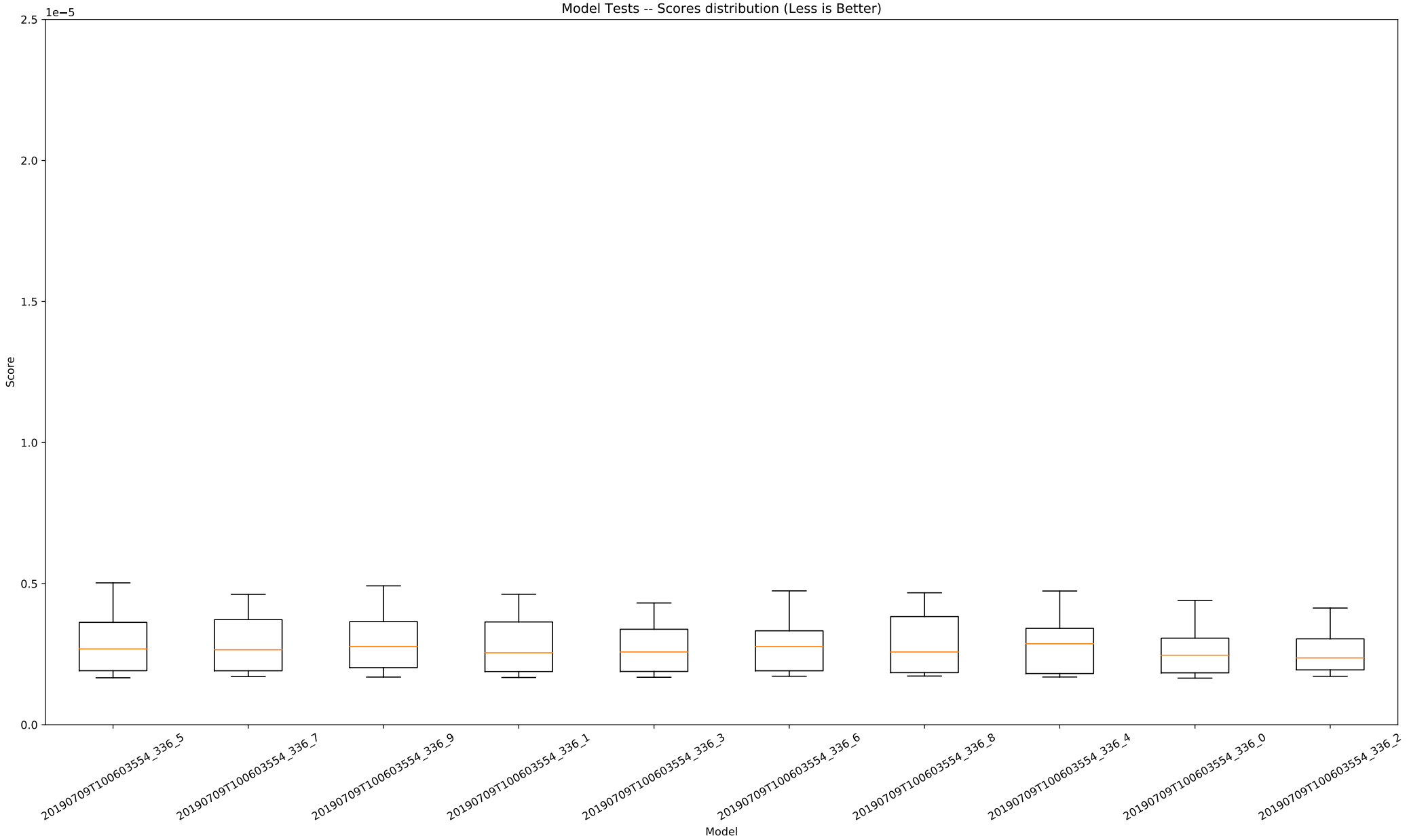

Figure C.5: Box-plot: network `20190709T100603554_336` `anti-phototaxis` test scores.

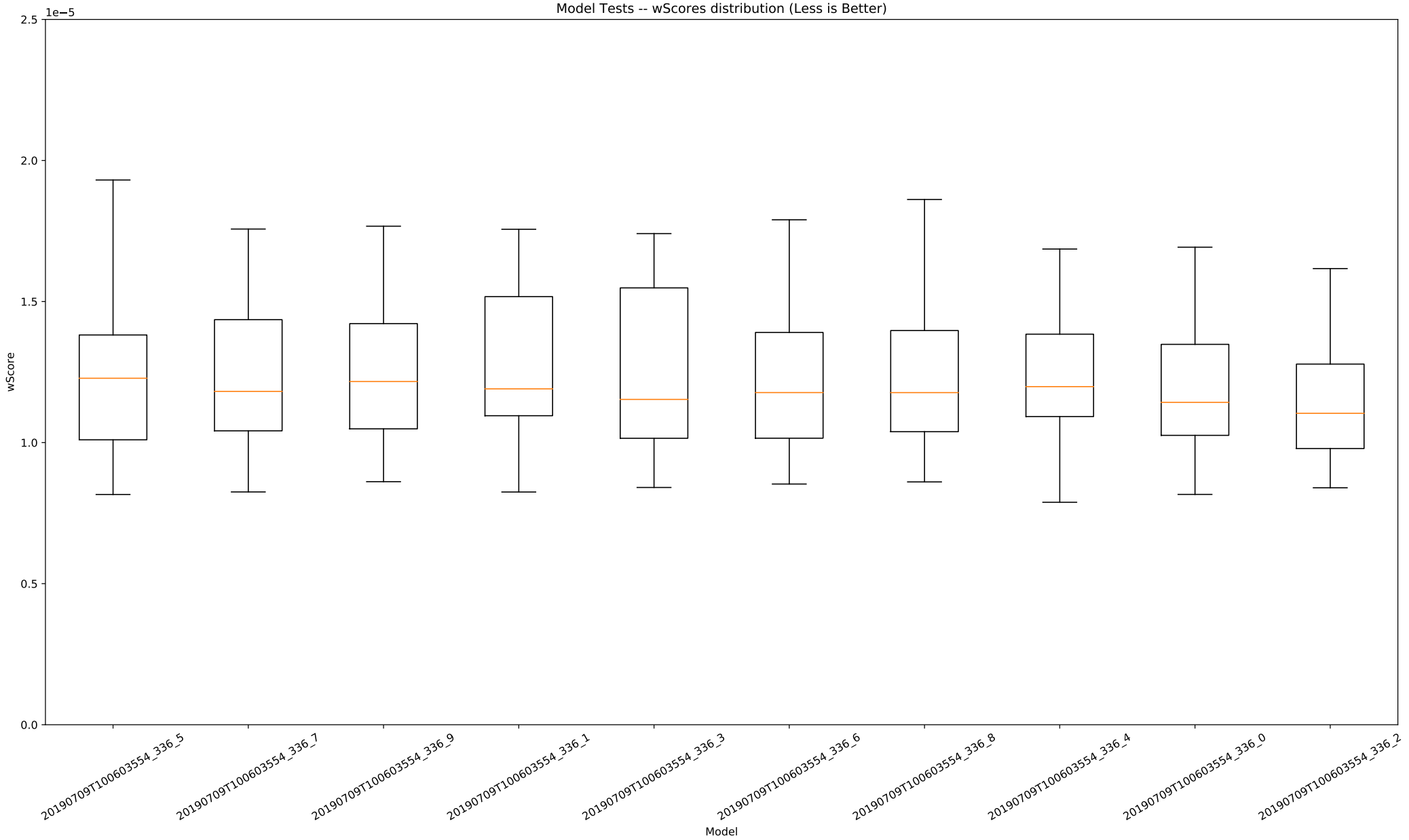

Figure C.6: Box-plot: network `20190709T100603554_336` weighted `anti-phototaxis` test scores.

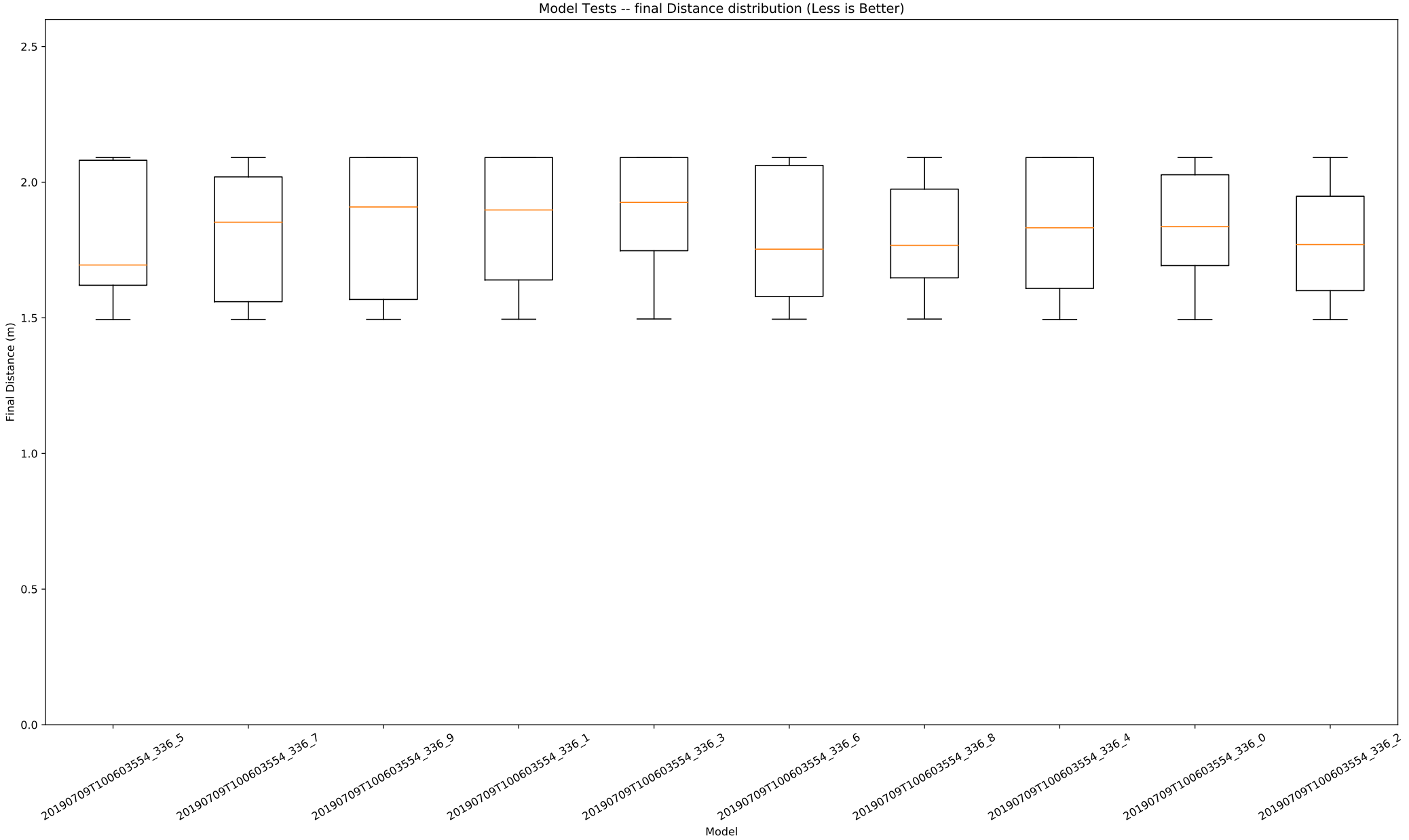

Figure C.7: Box-plot: network `20190709T100603554_336` `anti-phototaxis` test final distance.

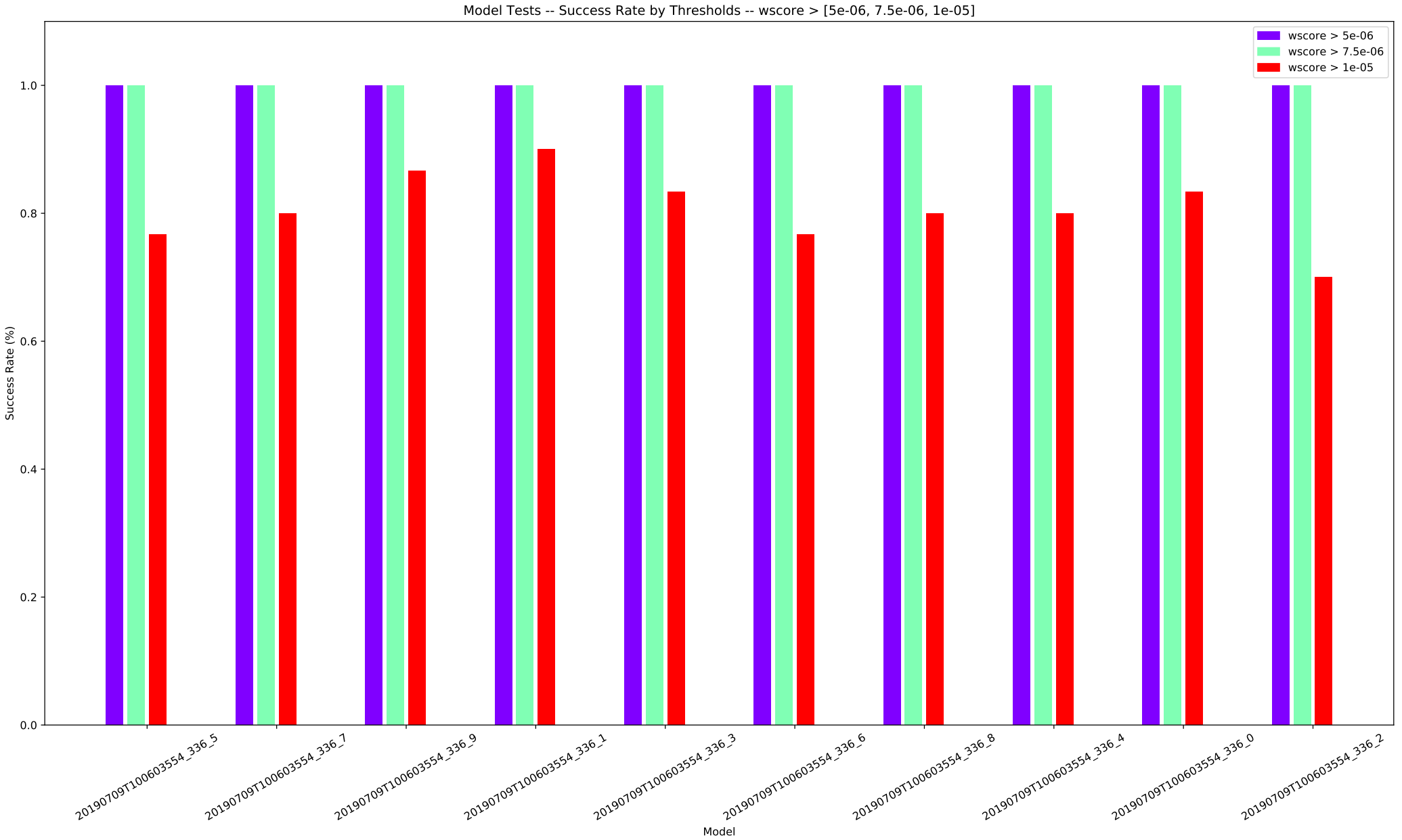

Figure C.8: Bar-plot: network `20190709T100603554_336`, each instance `anti-phototaxis` scores are tested against a set of optimal scores.

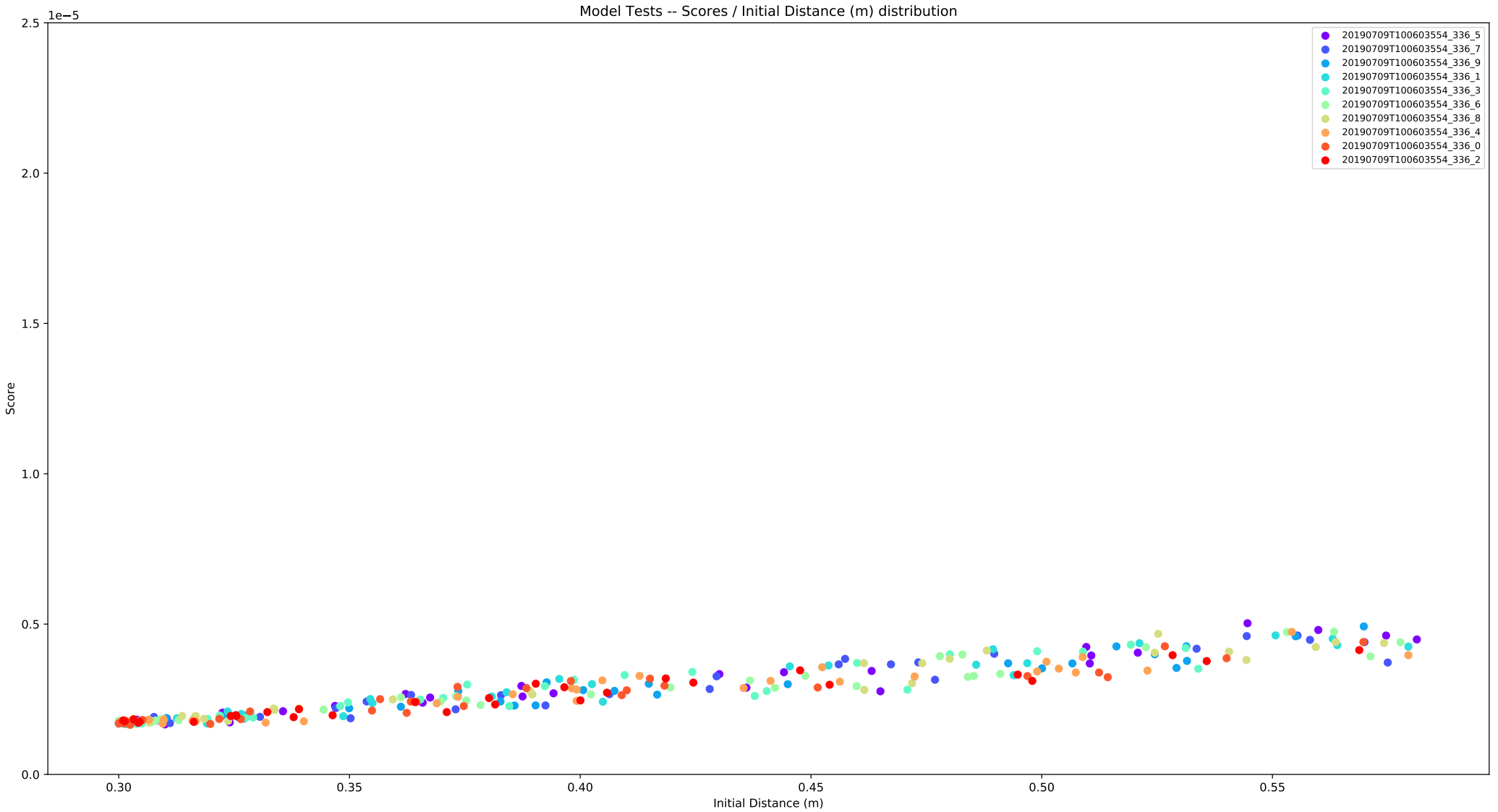

Figure C.9: Scatter-plot: network `20190709T100603554_336`, x-axis → initial distance, y-axis → `anti-phototaxis` scores

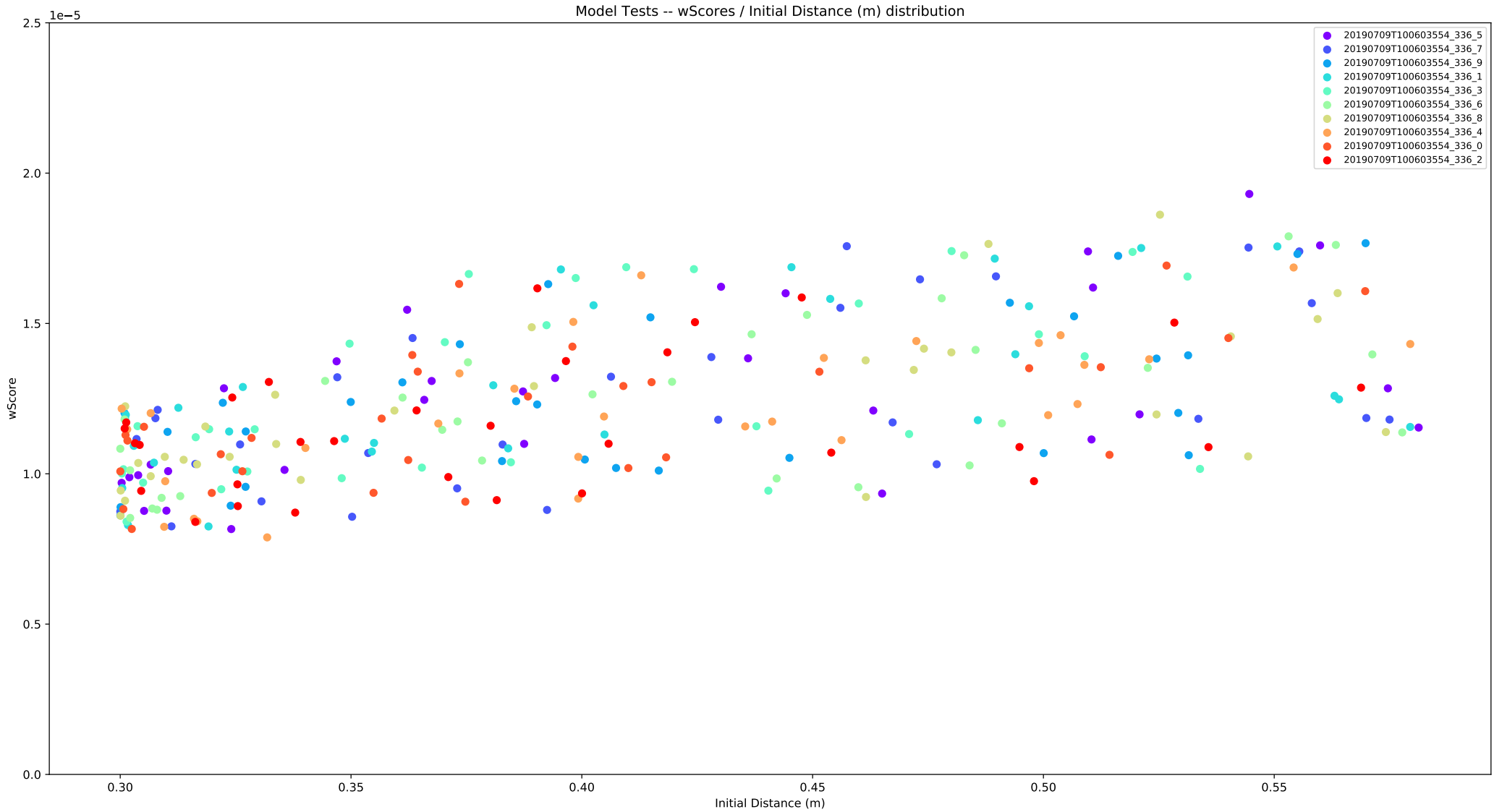

Figure C.10: Scatter-plot: network 20190709T100603554_336, x-axis → initial distance, y-axis → `anti-phototaxis` scores

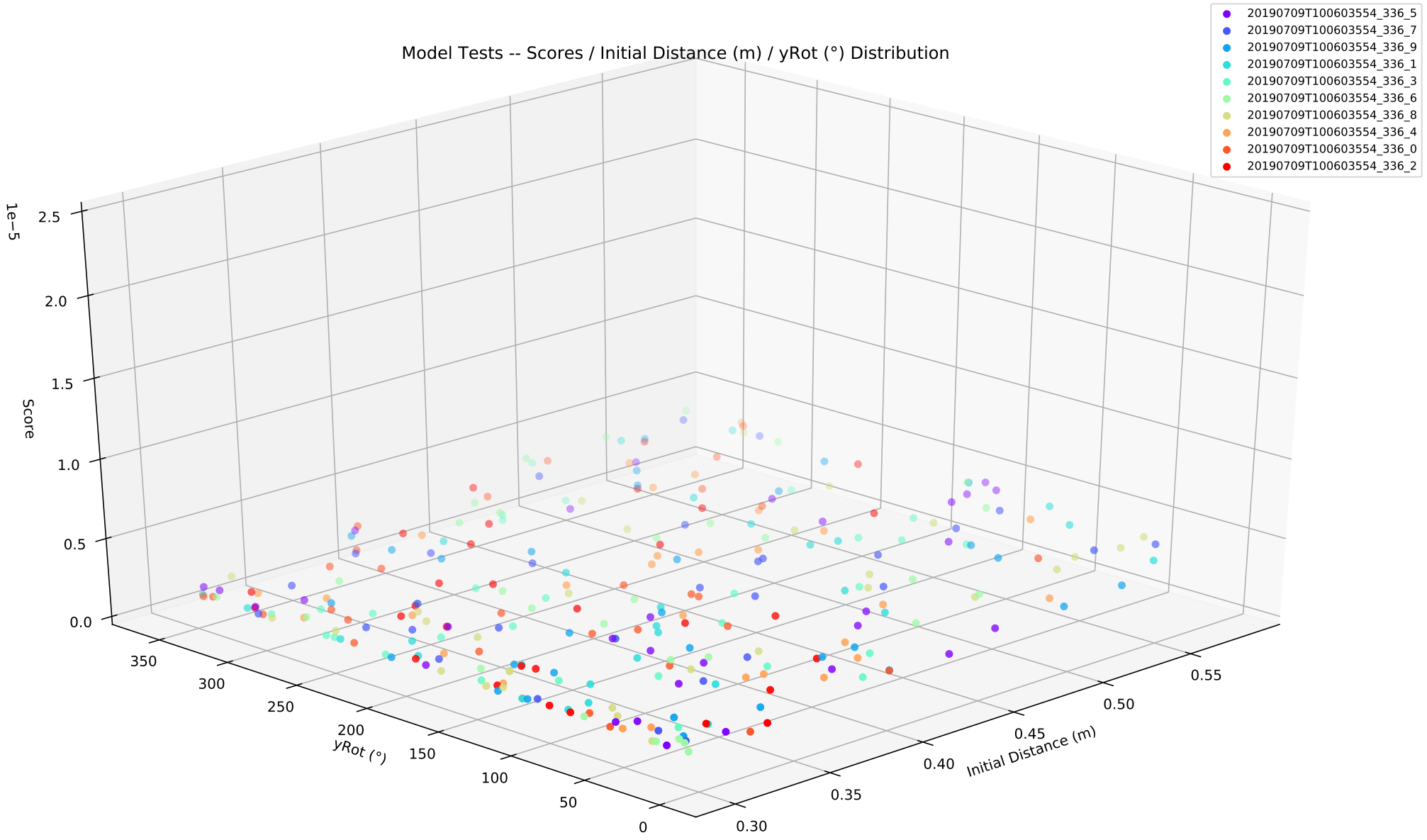

Figure C.11: Scatter-plot: network `20190709T100603554_336`, x-axis → initial distance, y-axis → agent rotation, z-axis → `anti-phototaxis` scores

Figure C.12: Scatter-plot: network 20190709T100603554_336, x-axis → initial distance, y-axis → agent rotation, z-axis → `anti-phototaxis` weighted scores

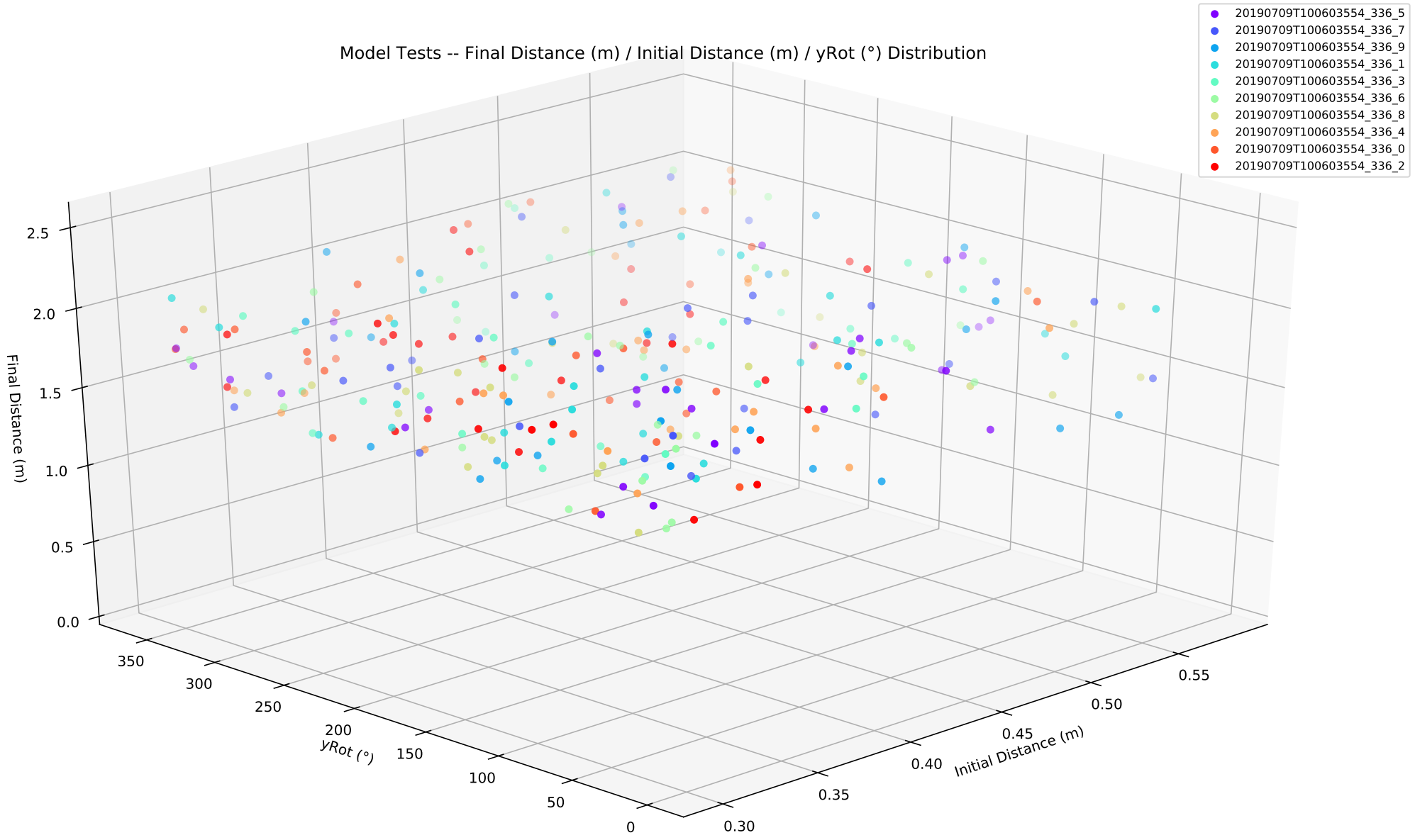

Figure C.13: Scatter-plot: network `20190709T100603554_336`, x-axis → initial distance, y-axis → agent rotation, z-axis → `anti-phototaxis` final distance

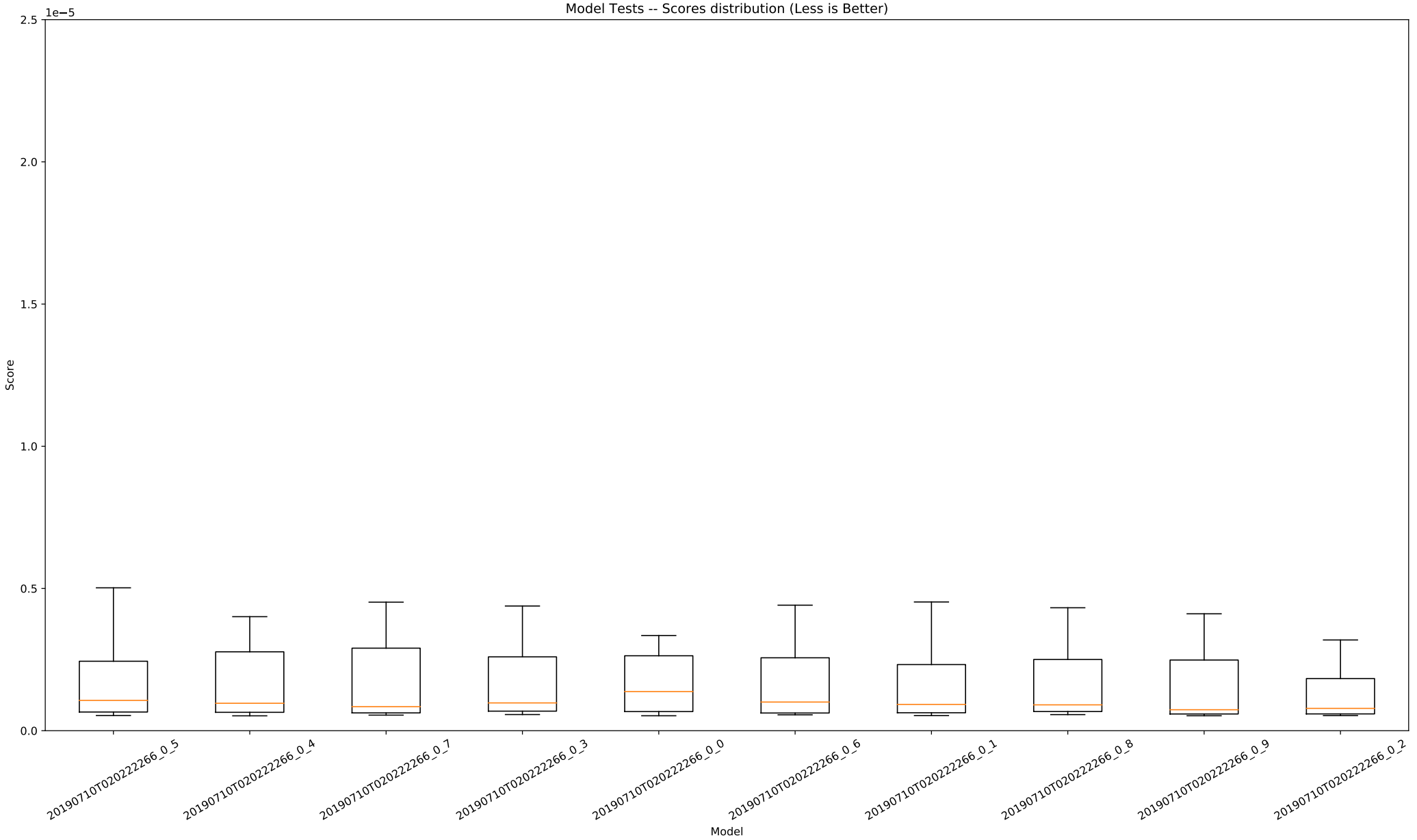

Figure C.14: Box-plot: network `20190710T020222266` `anti-phototaxis` test scores.

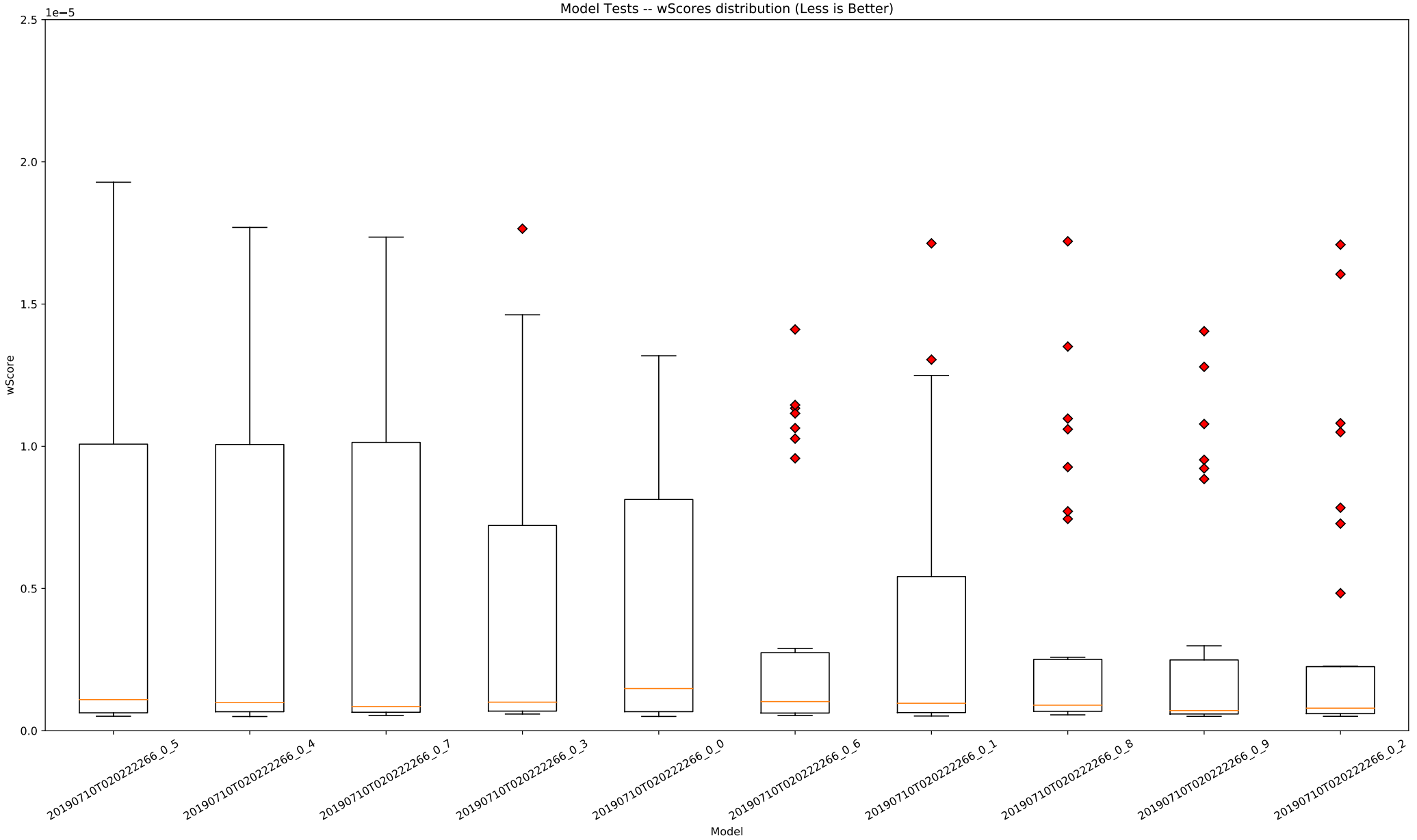

Figure C.15: Box-plot: network 20190710T020222266 weighted `anti-phototaxis` test scores.

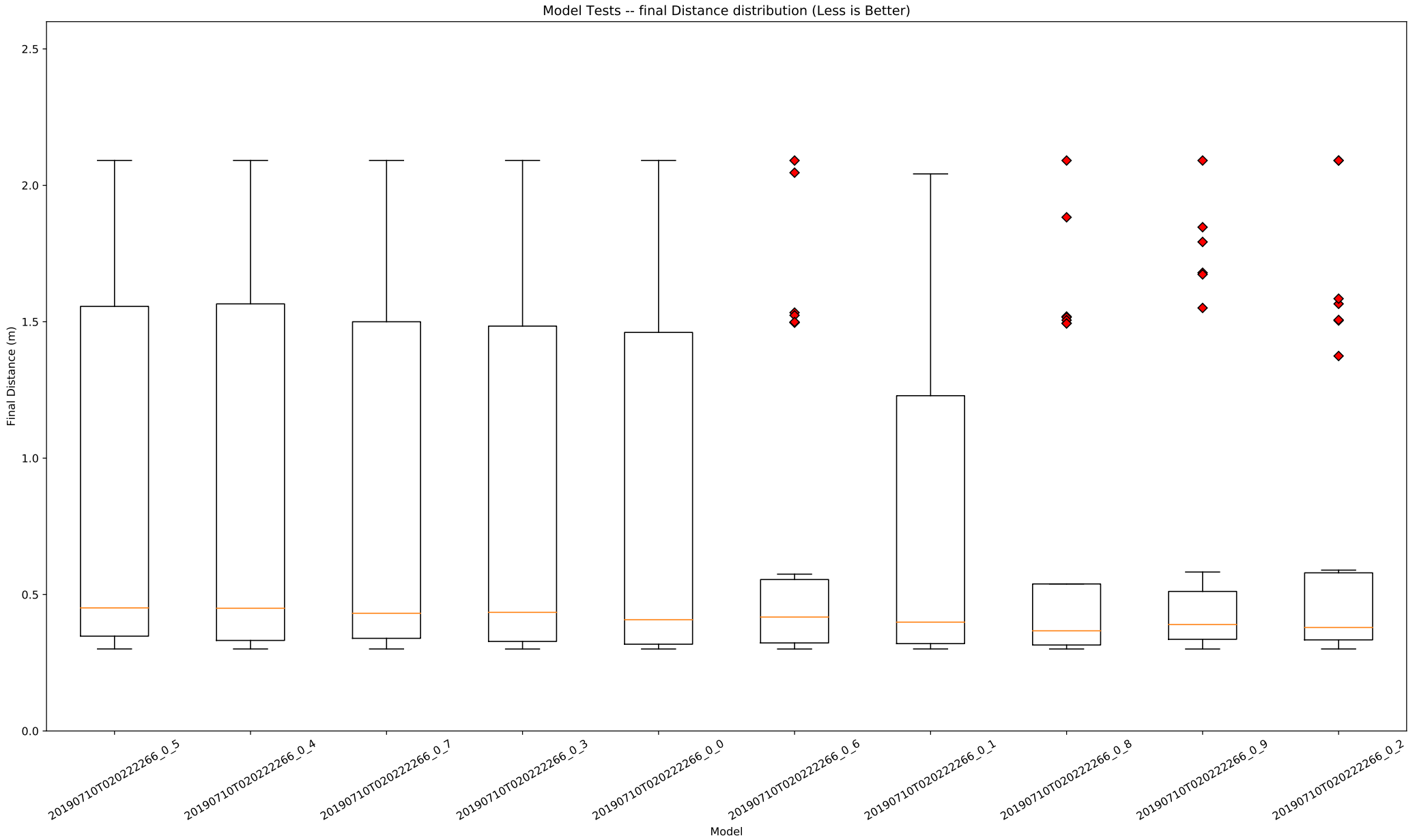

Figure C.16: Box-plot: network `20190710T020222266` `anti-phototaxis` test final distances.

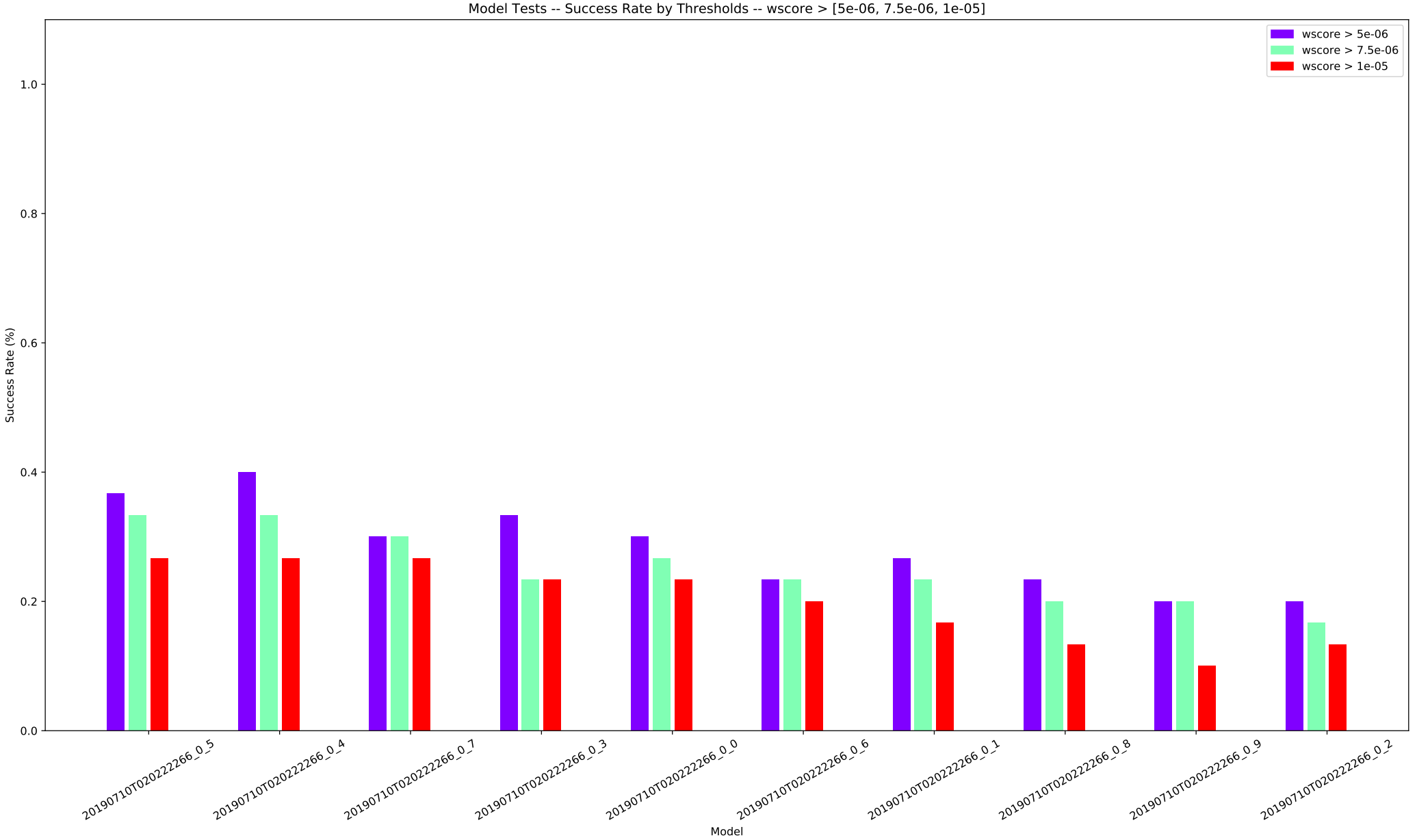

Figure C.17: Bar-plot: network `20190710T020222266`, each network `anti-phototaxis` score is tested against a set of optimal scores.

Appendix D

Selective Boolean Network Controller

In this appendix are collected all the figures regarding the testing of a `Selective Boolean Network Controller` expressing both `phototaxis` and `anti-phototaxis` as an effect of a differentiation process after a *noise phase*, like explained in 3.2.3. In order to display them correctly and to ease the view experience, the figures are proposed on a whole page scale, with an orientation from inside toward outside.

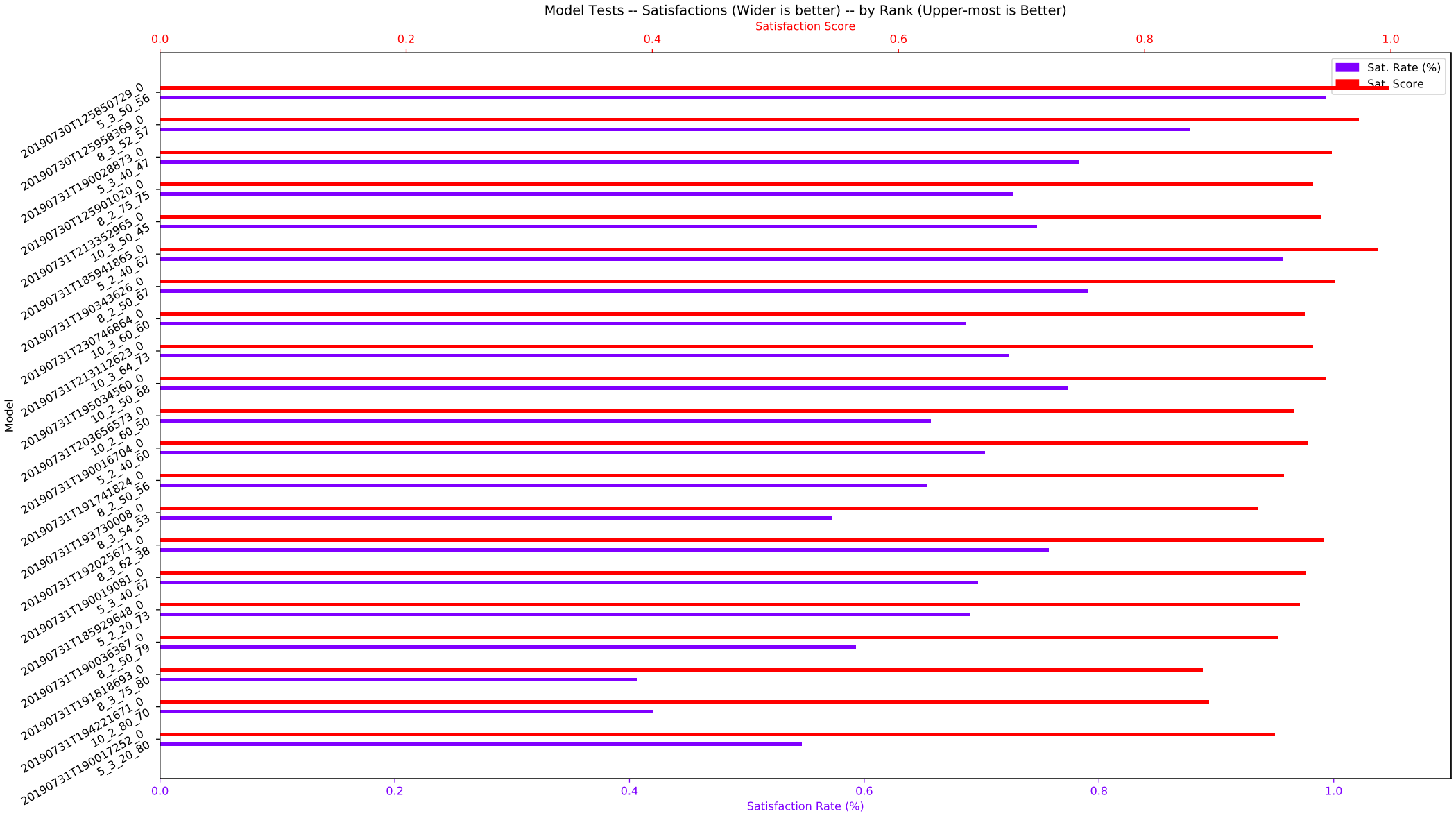

Figure D.1: Bar-plot: each network global *satisfaction rate* and *satisfaction score*.

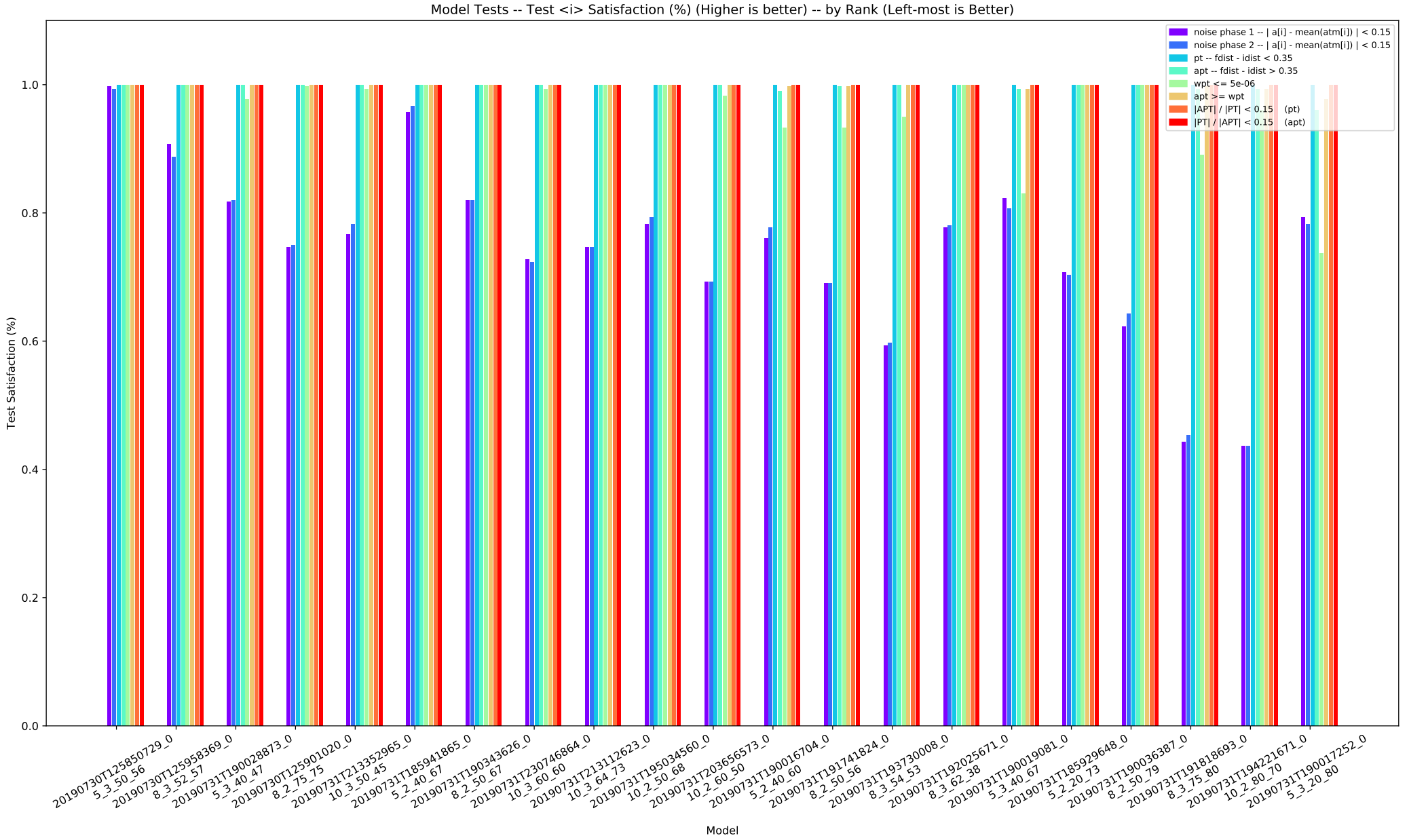

Figure D.2: Bar-plot: each network satisfaction percentage for each of the eight boolean constraints $b_i \in B$.

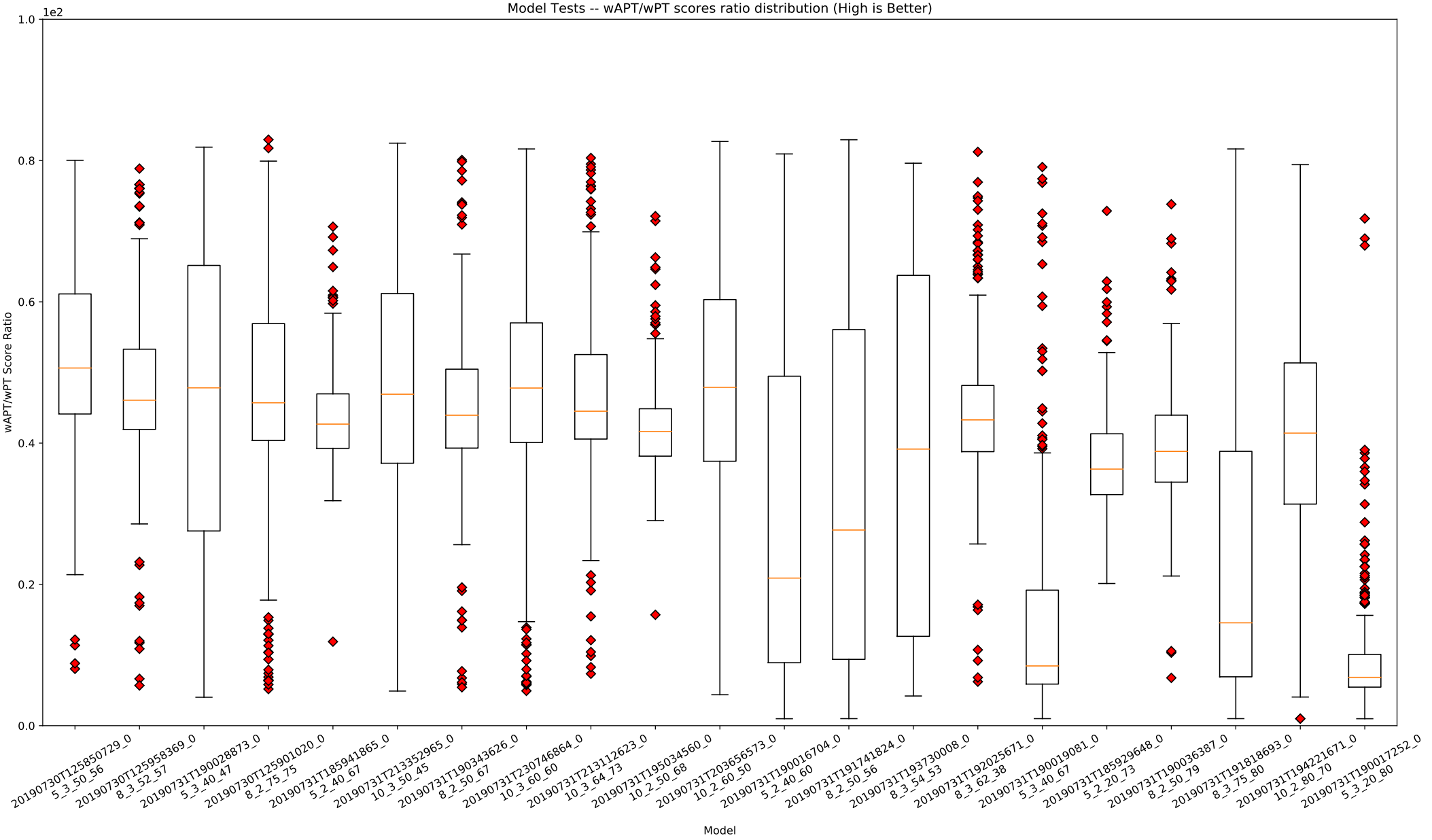

Figure D.3: Box-plot: each network weighted `anti-phototaxis` and `phototaxis` score ratio.

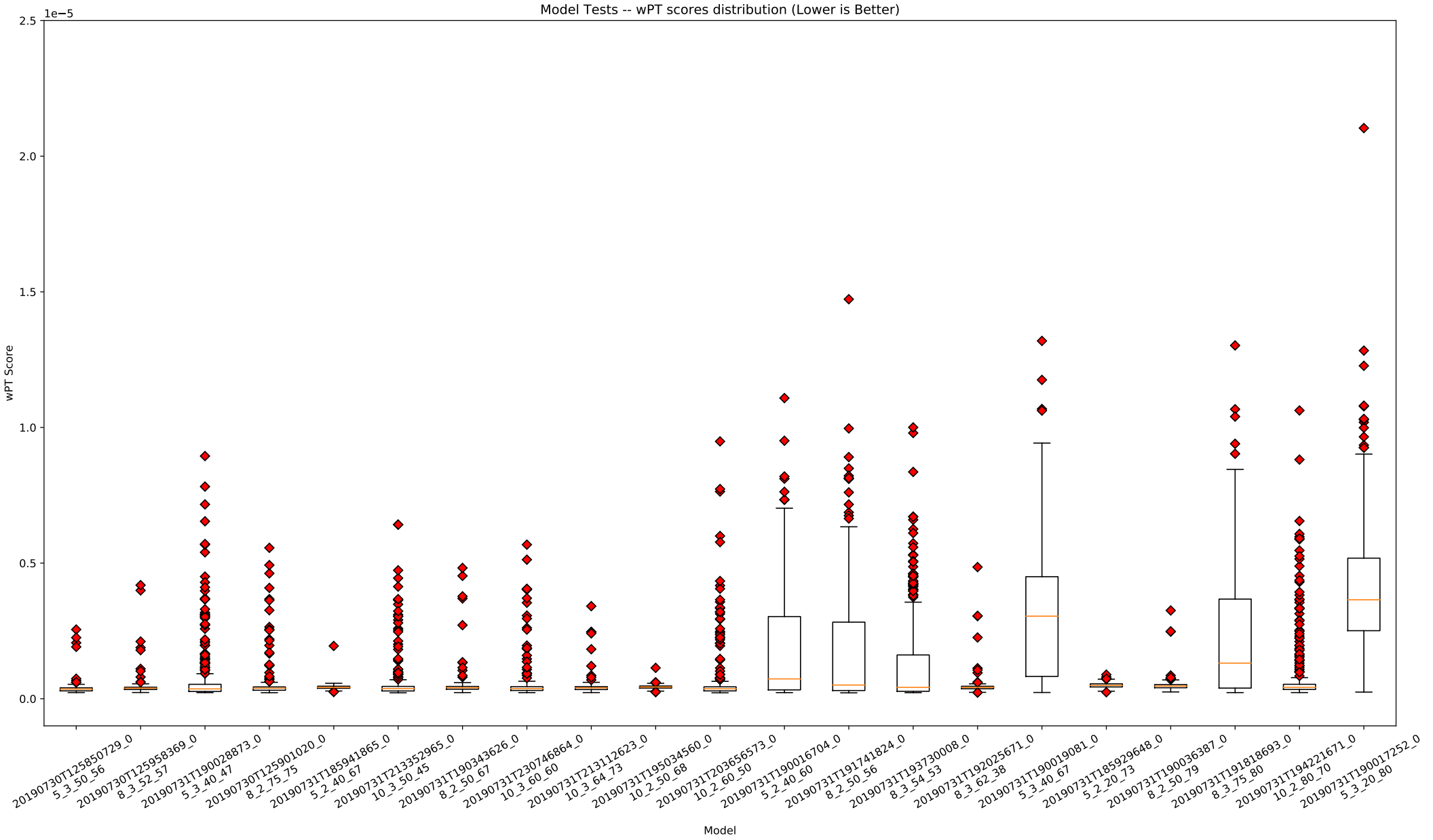

Figure D.4: Box-plot: each network weighted `phototaxis` scores achieved during the testing phase.

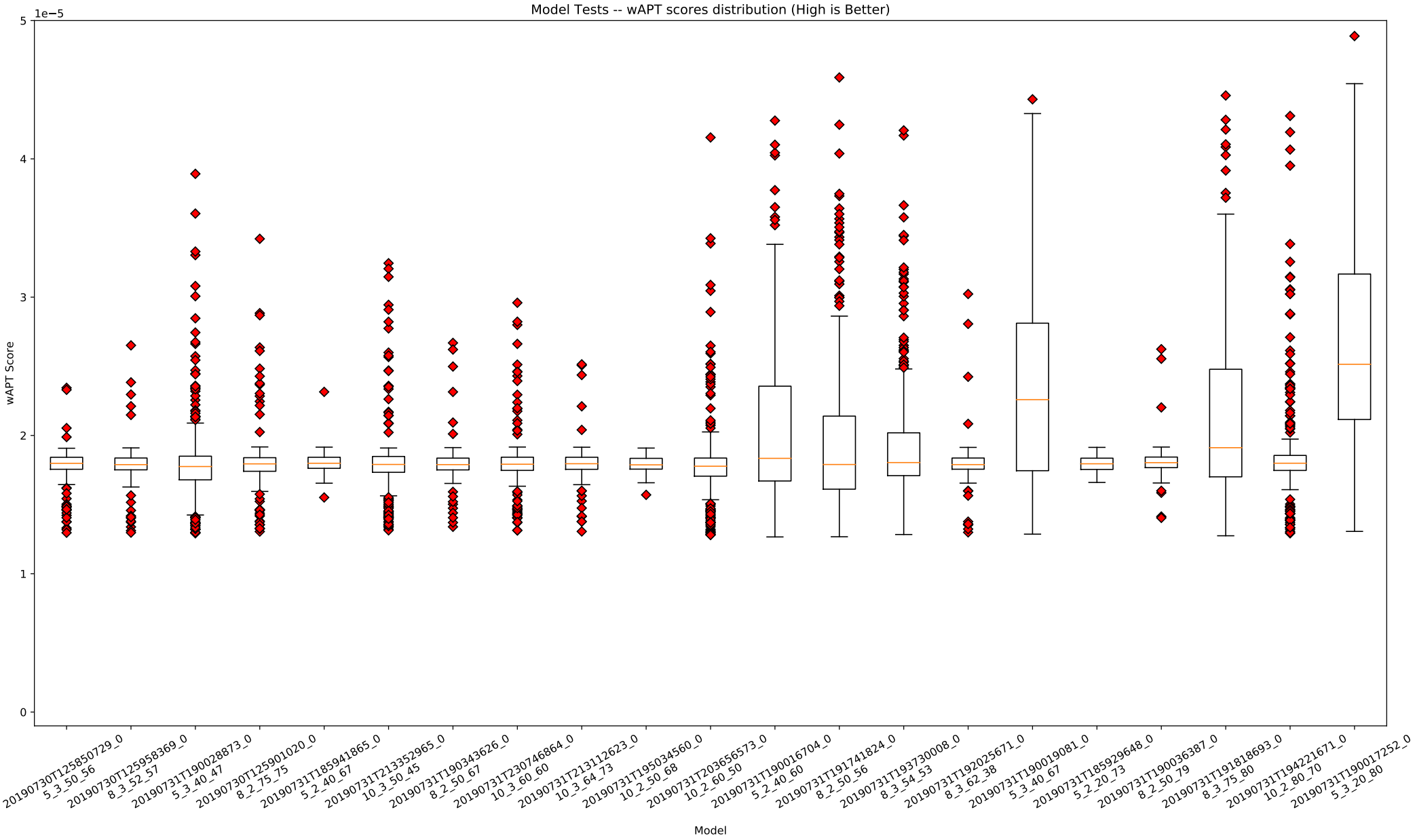

Figure D.5: Box-plot: each network weighted `anti-phototaxis` scores achieved during the testing phase.

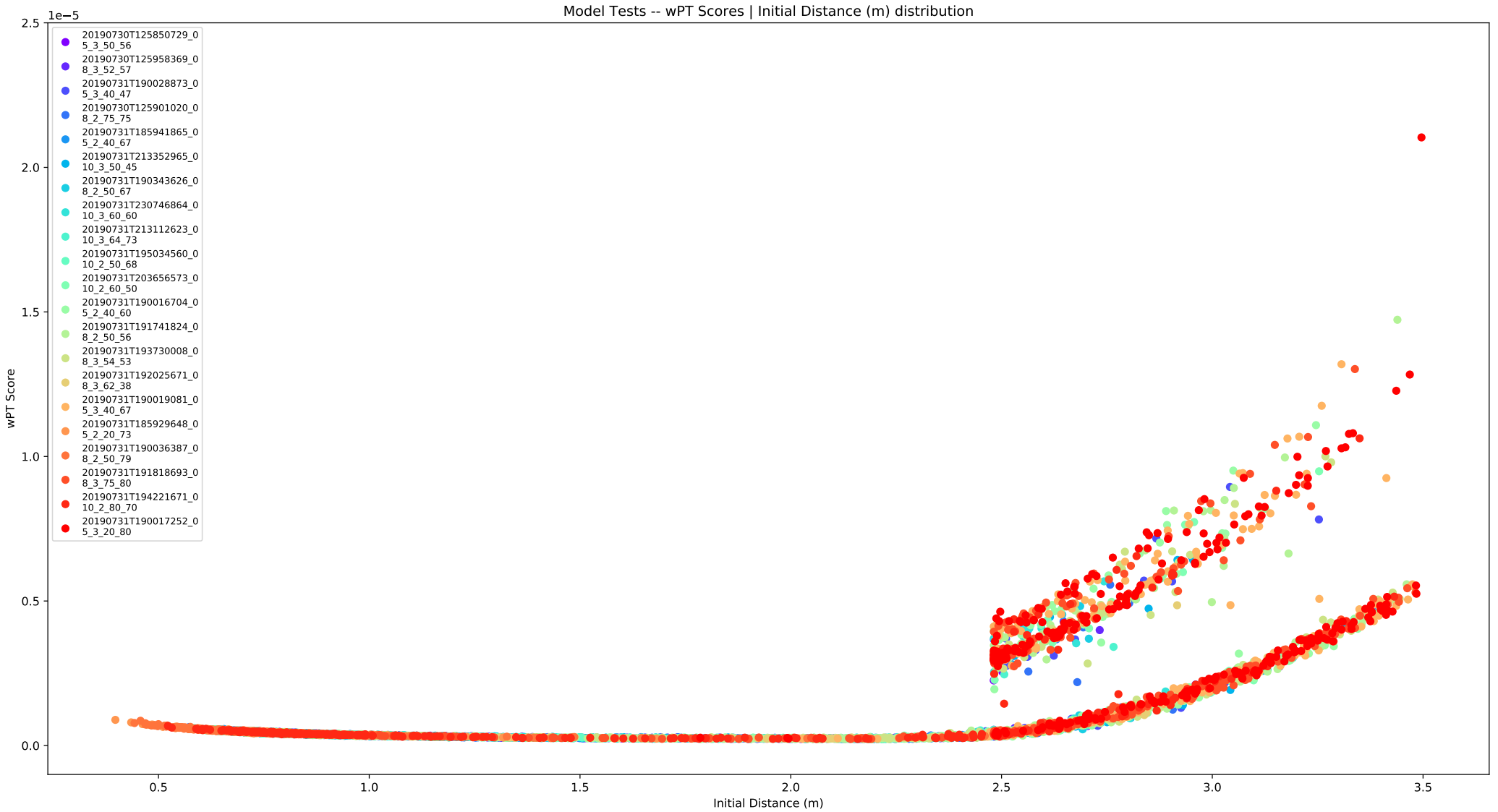

Figure D.6: Scatter-plot: all networks, x-axis → initial distance, y-axis → weighted `phototaxis` scores.

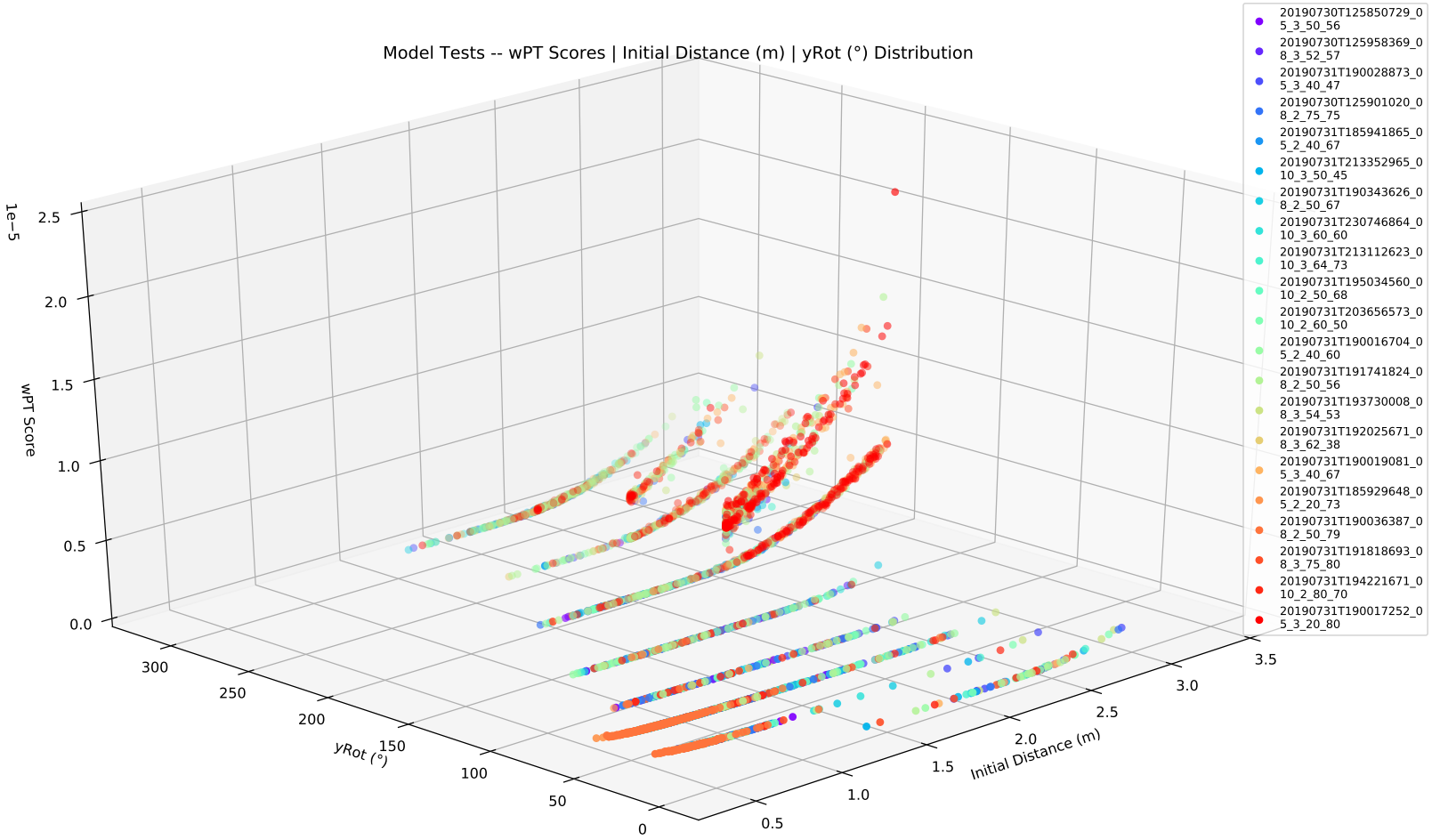

Figure D.7: Scatter-plot: all networks, x-axis → initial distance, y-axis → initial rotation, z-axis → weighted `phototaxis` scores.

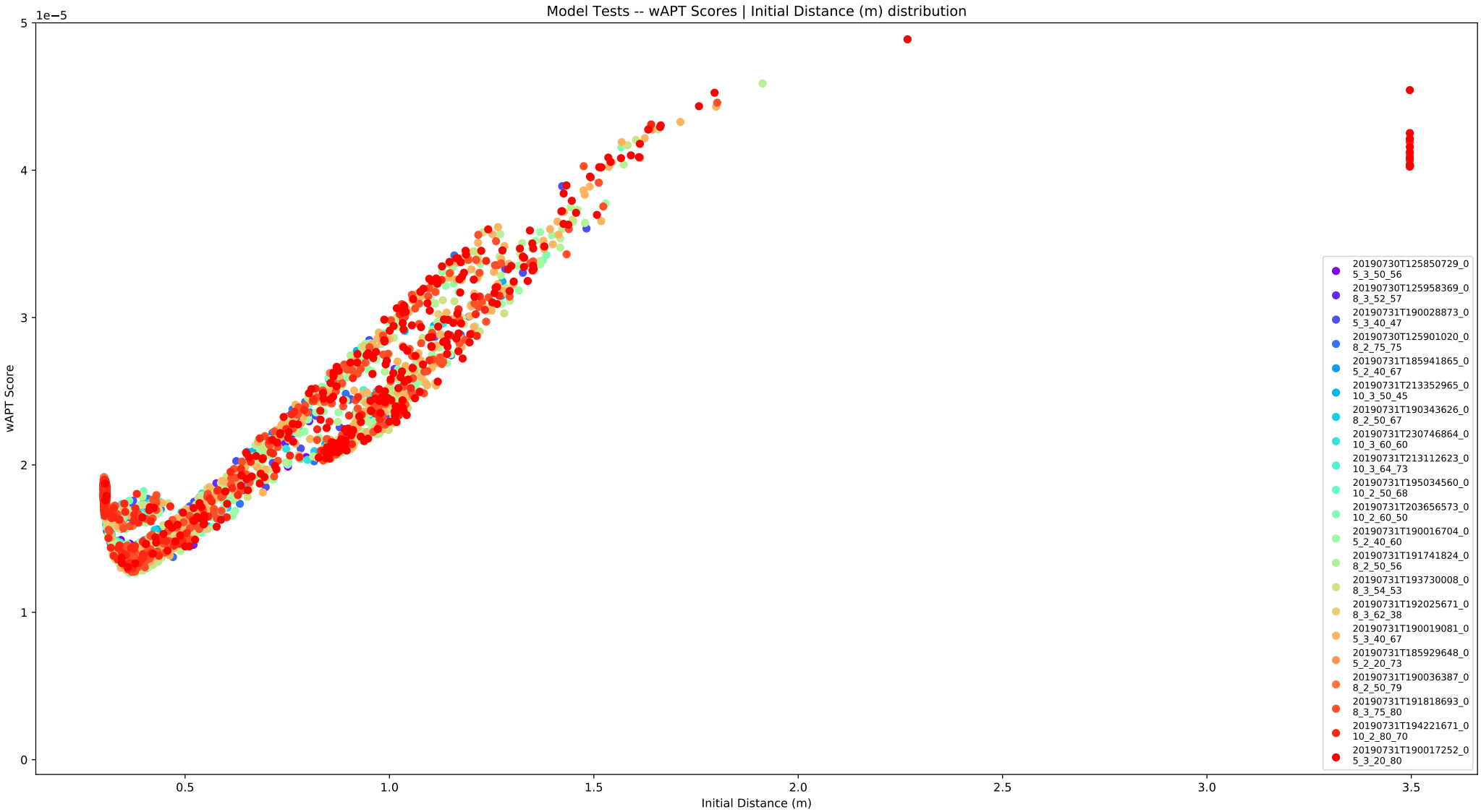

Figure D.8: Scatter-plot: all networks, x-axis → initial distance, y-axis → weighted `anti-phototaxis` scores.

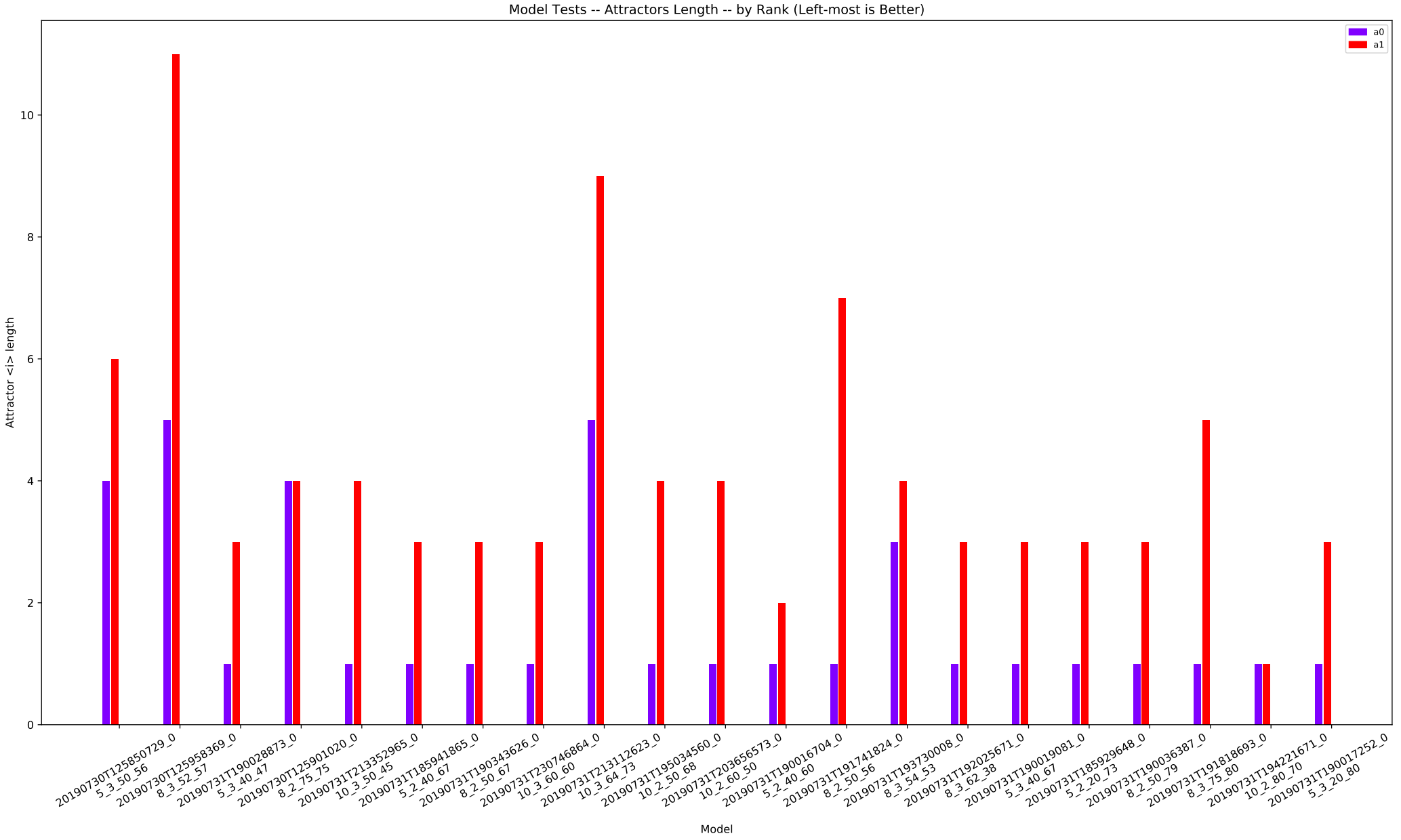

Figure D.9: Attractors length for each network.

Bibliography

[1] Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution*. OUP USA, 1993.

[2] Marco Villani and Roberto Serra. On the dynamical properties of a model of cell differentiation. *EURASIP Journal on Bioinformatics and Systems Biology*, 2013(1):4, 2013.

[3] Andrea Roli and Michele Braccini. Attractor landscape: A bridge between robotics and synthetic biology. *cell differentiation*, 4:5, 2018.

[4] Andre S Ribeiro and Stuart A Kauffman. Noisy attractors and ergodic sets in models of gene regulatory networks. *Journal of theoretical biology*, 247(4):743–755, 2007.

[5] CHRISTOPH Fretter and BARBARA Drossel. Response of boolean networks to perturbations. *The European Physical Journal B*, 62(3):365–371, 2008.

[6] Andrea Roli, Mattia Manfroni, Carlo Pinciroli, and Mauro Birattari. On the design of boolean network robots. In *European Conference on the Applications of Evolutionary Computation*, pages 43–52. Springer, 2011.

[7] Andrea Roli, Marco Villani, Roberto Serra, Stefano Benedettini, Carlo Pinciroli, and Mauro Birattari. Dynamical properties of artificially evolved boolean network robots. In *Congress of the Italian Association for Artificial Intelligence*, pages 45–57. Springer, 2015.

[8] Sui Huang, Ingemar Ernberg, and Stuart Kauffman. Cancer attractors: a systems view of tumors from a gene network dynamics and developmental perspective. In *Seminars in cell & developmental biology*, volume 20, pages 869–876. Elsevier, 2009.

[9] Michele Braccini. Applications of biological cell models in robotics. *arXiv preprint arXiv:1712.02303*, 2017.

[10] Peter Eggenberger. Cell interactions as a control tool of developmental processes for evolutionary robotics. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 440–448, 1996.

[11] Olivier Michel and Joëlle Biondi. Morphogenesis of neural networks. *Neural Processing Letters*, 2(1):9–12, 1995.

[12] Olivier Michel and Joëlle Biondi. From the chromosome to the neural network. In *Artificial Neural Nets and Genetic Algorithms*, pages 80–83. Springer, 1995.

[13] Peter Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the fourth european conference on Artificial Life*, pages 205–213, 1997.

[14] Josh C Bongard and Rolf Pfeifer. Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The new species*, pages 237–258. Springer, 2003.

[15] U Bastolla and GIORGIO Parisi. A numerical study of the critical line of kauffman networks. *Journal of theoretical biology*, 187(1):117–133, 1997.

[16] Michele Braccini, Andrea Roli, Marco Villani, and Roberto Serra. Automatic design of boolean networks for cell differentiation. In *Italian Workshop on Artificial Life and Evolutionary Computation*, pages 91–102. Springer, 2016.

[17] Michele Braccini. *Automatic design of boolean networks for modelling differentiation trees*. PhD thesis, Alma Mater Studiorum Università di Bologna, Campus of Cesena, 2016.

[18] Sui Huang. The molecular and mathematical basis of waddington's epigenetic landscape: A framework for post-darwinian biology? *Bioessays*, 34(2):149–157, 2012.

[19] Marco Villani, Alessia Barbieri, and Roberto Serra. A dynamical model of genetic networks for cell differentiation. *PloS one*, 6(3):e17703, 2011.

[20] Olivier Michel. Cyberbotics ltd. webots^{TM}: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004.

[21] Lorenzo Garattoni, Andrea Roli, Matteo Amaducci, Carlo Pinciroli, and Mauro Birattari. Boolean network robotics as an intermediate step in the synthesis of finite state machines for robot control. In *Artificial Life Conference Proceedings 13*, pages 783–790. MIT Press, 2013.

[22] Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. MIT press, 2001.

[23] Randall D Beer. A dynamical systems perspective on agent-environment interaction. *Artificial intelligence*, 72(1-2):173–215, 1995.

[24] Stefano Benedettini, Marco Villani, Andrea Roli, Roberto Serra, Mattia Manfroni, Antonio Gagliardi, Carlo Pinciroli, and Mauro Birattari. Dynamical regimes and learning properties of evolved boolean networks. *Neurocomputing*, 99:111–123, 2013.

[25] Roberto Serra, Marco Villani, Alessia Barbieri, Stuart A Kauffman, and Annamaria Colacci. On the dynamics of random boolean networks subject to noise: attractors, ergodic sets and cell types. *Journal of theoretical biology*, 265(2):185–193, 2010.