TurnitinUK Originality Report

Automated Reverse Engineering of Agent Behaviors by Wei Li

From PhD Thesis Final Submission (ACS6220 Presentation of Control Engineering Research (GRADUATE YEAR 2014~15))

Similarity Index	Similarity by Source	
22%	Internet Sources:	16%
	Publications:	14%
	Student Papers:	10%

Processed on 30-Sep-2015 2:02 PM BST
ID: 46960606
Word Count: 45162

sources:

1 5% match (Internet from 08-Dec-2014)
http://naturalrobotics.group.shef.ac.uk/publications/2014-gecco-li.pdf

2 4% match (Internet from 08-Dec-2014)
http://naturalrobotics.group.shef.ac.uk/publications/2013-gecco-li.pdf

3 4% match (student papers from 03-Dec-2012)
Class: ACS6220 Presentation of Control Engineering Research (GRADUATE YEAR 2012~13)
Assignment:
Paper ID: 19755858

4 1% match (Internet from 30-Sep-2015)
http://naturalrobotics.group.shef.ac.uk/supp/2015-002/

5 1% match (student papers from 30-Sep-2014)
Submitted to University of Sheffield on 2014-09-30

6 1% match (student papers from 01-Dec-2012)
Class: ACS6220 Presentation of Control Engineering Research (GRADUATE YEAR 2012~13)
Assignment:
Paper ID: 19680986

7 < 1% match (publications)
Li, Wei, Melvin Gauci, and Roderich Gross. "Coevolutionary learning of swarm behaviors without metrics", Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO 14, 2014.

8 < 1% match (publications)
Gauci, M., J. Chen, W. Li, T. J. Dodd, and R. Gross. "Self-organized aggregation without computation", The International Journal of Robotics Research, 2014.

9 < 1% match (publications)
Li, Wei, Melvin Gauci, and Roderich Gross. "A coevolutionary approach to learn animal behavior through controlled interaction", Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO 13, 2013.

10 < 1% match (publications)
Springer Handbook of Computational Intelligence, 2015.

11 < 1% match (student papers from 29-Sep-2014)
Submitted to University of Sheffield on 2014-09-29

12 < 1% match (publications)
Yan, Xinan, Alei Liang, and Haibing Guan. "An algorithm for self-organized aggregation of swarm robotics using timer", 2011 IEEE Symposium on Swarm Intelligence, 2011.

13 < 1% match (publications)

J. C. Ferreri. "Engineering Judgment and Natural Circulation Calculations", Science and Technology of Nuclear Installations, 2011

14 < 1% match (Internet from 28-Mar-2010)
http://laral.istc.cnr.it/trianni/docs/ieee-smc-b.pdf

15 < 1% match (Internet from 04-Jul-2014)
http://riunet.upv.es/bitstream/handle/10251/16353/Proceedings%20AAMAS%202012.pdf.txt?sequence=2

16 < 1% match (Internet from 17-Aug-2015)
http://followscience.com/content/515970/nonlinear-system-identification-a-state-space-approach/

17 < 1% match (student papers from 01-Sep-2013)
Submitted to University of Sheffield on 2013-09-01

18 < 1% match (student papers from 01-Sep-2015)
Submitted to University of Sheffield on 2015-09-01

19 < 1% match (student papers from 13-Jan-2014)
Submitted to Dhirubhai Ambani Institute of Information and Communication on 2014-01-13

20 < 1% match (Internet from 17-Jul-2006)
http://gzam.networkrail.co.uk/GreenZoneGuides/EA_sat.pdf

21 < 1% match (publications)
Gong, Wenyin, Aimin Zhou, and Zhihua Cai. "A Multi-Operator Search Strategy based on Cheap Surrogate Models for Evolutionary Optimization", IEEE Transactions on Evolutionary Computation, 2015.

22 < 1% match (Internet from 09-Jan-2013)
http://www.aamas-conference.org/Proceedings/aamas2012/papers/4A_4.pdf

23 < 1% match (publications)
石 青. "Development of a Rat-like Robot and Its Applications in Animal Behavior Research", [出版者不明], 2013.

24 < 1% match (Internet from 18-Oct-2010)
http://iridia.ulb.ac.be/~rgross/publications/iros2009.pdf

25 < 1% match (Internet from 14-May-2012)
http://heikohamann.de/pub/hamannAlife12.pdf

26 < 1% match (Internet from 29-Oct-2014)
http://edoc.ub.uni-muenchen.de/14264/1/Pohl_Sebastian.pdf

27 < 1% match (publications)
Lopez-Franco, Carlos Villavicencio, Luis. "Image classification using PSO-SVM and an RGB-D sensor.(Research Article)(Report)", Mathematical Problems in Engineering, Annual 2014 Issue

28 < 1% match (student papers from 02-Apr-2015)
Submitted to Heriot-Watt University on 2015-04-02

29 < 1% match (Internet from 10-Oct-2014)
http://ww2.ii.uj.edu.pl/~tabor/prII09-10/perl/allarticles.html

30 < 1% match (Internet from 20-Feb-2015)
http://martin-ellis.net/thesis/phdthesis-ellis-2012.pdf

31 < 1% match (Internet from 07-Nov-2014)
http://www02.us.archive.org/stream/BiomedicalEngineeringHandbookVolumeIIi/BiomedicalEngineeringHandbookVolumeIIiJ.

32 < 1% match (student papers from 10-Aug-2014)
Submitted to National University of Singapore on 2014-08-10

33 < 1% match (Internet from 13-Sep-2014)
http://www.genetic-programming.org/hc2014/David-Paper.pdf

34 < 1% match (Internet from 02-Jun-2014)
http://sci2s.ugr.es/gbml/index.php

35 < 1% match (Internet from 18-Feb-2008)
http://www.site.uottawa.ca/~nsamaras/thesis/thesis_pdf.pdf

36 < 1% match (Internet from 18-Oct-2010)
http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2008-005r001.pdf

37 < 1% match (Internet from 08-Dec-2012)
http://publications.lib.chalmers.se/records/fulltext/144655.pdf

38 < 1% match (Internet from 03-Oct-2008)
http://www.cyber-s.ne.jp/Top/Volume/1-2/0008tf.pdf

39 < 1% match (publications)
Obuchowicz, Andrzej and Prętki, Przemysław. "Self-Adaptive Stable Mutation Based on Discrete Spectral Measure for Evolutionary Algorithms", Journal of Telecommunications & Information Technology, 2011.

40 < 1% match (publications)
Kerdprasop, Nittaya and Kerdprasop, Kittisak. "Knowledge Engineering Process for a Rapid Prototyping of Inductive Expert System", International Journal of Computer Science Issues (IJCSI), 2013.

41 < 1% match (Internet from 12-Dec-2014)
http://nelsonrobotics.org/publications.html

42 < 1% match (Internet from 26-Sep-2010)
http://www.ri.cmu.edu/pub_files/2009/10/SMC.pdf

43 < 1% match (Internet from 09-Jan-2015)
http://cs.cug.edu.cn/teacherweb/lichanghe/papers/Journals/IEEETSMCB/IEEETSMCB11.pdf

44 < 1% match (Internet from 17-Dec-2007)
http://www-timc.imag.fr/Olivier.Francois/design01.pdf

45 < 1% match (Internet from 26-Sep-2011)
http://people.cs.uu.nl/dejong/publications/parnashgecco06.pdf

46 < 1% match (Internet from 28-Oct-2010)
http://arxiv.org/pdf/0704.1147

47 < 1% match ()

http://www.librecours.org/documents/8/813.pdf

48 < 1% match (publications)
Amanda J. C. Sharkey. "Swarm robotics and minimalism", Connection Science, 9/2007

49 < 1% match (publications)
Leonce, Armand, Bryan Hoke, and Dean F. Hougen. "Evolution of robot-to-robot nurturing and nurturability", 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL), 2012.

50 < 1% match (publications)
Abdul Rahman Hafiz Abdul Ghani. "Bio-Inspired Active Robot Vision Toward Understanding of the Cortical Mechanism", University of Fukui, 2014.

51 < 1% match (student papers from 11-Nov-2012)
Submitted to KTH - The Royal Institute of Technology on 2012-11-11

52 < 1% match (Internet from 12-Sep-2014)
http://www.mtome.com/Publications/JMLR/jmlr-vol6-partB.pdf

53 < 1% match (Internet from 10-Oct-2014)
http://robobrarian.info/b_pubs/IROS10-Smith.pdf

54 < 1% match (publications)
Kouno, Asuki, Jean-Marc Montanier, Shigeru Takano, Nicolas Bredeche, Marc Schoenauer, Michèle Sebag, and Einoshin Suzuki. "On-Board Evolutionary Algorithm and Off-Line Rule Discovery for Column Formation in Swarm Robotics", 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, 2011.

55 < 1% match (publications)
Mariani, Leonardo, Oliviero Riganelli, Mauro Santoro, and Muhammad Ali. "G-RankTest: Regression testing of controller applications", 2012 7th International Workshop on Automation of Software Test (AST), 2012.

56 < 1% match (student papers from 27-May-2014)
Submitted to University of Surrey on 2014-05-27

57 < 1% match (student papers from 07-Apr-2005)
Submitted to National University of Singapore on 2005-04-07

58 < 1% match (student papers from 22-Aug-2011)
Submitted to University of Oxford on 2011-08-22

59 < 1% match (Internet from 30-Sep-2009)
http://www.demo.cs.brandeis.edu/papers/bucci_diss.pdf

60 < 1% match (Internet from 20-Aug-2014)
http://lmarti.com/wp-content/uploads/2014/07/marti-et-al-2010-model-building-tech-rep.pdf

61 < 1% match (Internet from 20-Sep-2009)
http://ieeexplore.ieee.org/xpl/absprintf.jsp?arnumber=1605376&page=FREE

62 < 1% match (Internet from 26-Sep-2010)
http://bbdl.usc.edu/Papers/Valero-Cuevas_et_al_RBME_Modeling_2009.pdf

63 < 1% match (Internet from 30-Mar-2012)
http://www.cis.upenn.edu/~cjtaylor/PUBLICATIONS/pdfs/CowleyTaylorSouthallICRA11.pdf

64 < 1% match (publications)
Sylvain Koos. "Automatic system identification based on coevolution of models and tests", 2009 IEEE Congress on Evolutionary Computation, 05/2009

65 < 1% match (publications)
Tina Yu. "Coevolution of simulator proxies and sampling strategies for petroleum reservoir modeling", 2009 IEEE Congress on Evolutionary Computation, 05/2009

66 < 1% match (student papers from 02-Dec-2011)
Submitted to University of Sheffield on 2011-12-02

67 < 1% match (student papers from 21-Aug-2015)
Submitted to Cranfield University on 2015-08-21

68 < 1% match (Internet from 07-Jan-2010)
http://www.cs.uvm.edu/~jbongard/papers/2008_TEC_Bongard.pdf

69 < 1% match (Internet from 27-Nov-2010)
http://media.wiley.com/product_data/excerpt/12/04716695/0471669512.pdf

70 < 1% match (Internet from 25-Oct-2010)
http://www.iguanaworks.net/~shucker/papers/acc2007.pdf

71 < 1% match (Internet from 22-Feb-2012)
http://www.shuguangli.com/OTHERS/CEC2011-Li.pdf

72 < 1% match (Internet from 16-Oct-2011)
http://www.cems.uvm.edu/~jbongard/papers/2009_ICDM_Lu.pdf

73 < 1% match (Internet from 04-Dec-2014)
http://www.tdx.cat/bitstream/handle/10803/6674/TJAGL1de1.pdf.txt?sequence=2

74 < 1% match (Internet from 29-Dec-2012)
http://metahack.org/enon-journal.pdf

75 < 1% match (Internet from 04-Sep-2011)
http://www.citeulike.org/user/hmedal/tag/supply

76 < 1% match (Internet from 19-Feb-2012)
http://www.olliebown.com/files/papers/bown_smc09.pdf

77 < 1% match (Internet from 16-Aug-2009)
http://web.hcsps.sa.edu.au/projects/flight/students/callum/bibliography.htm

78 < 1% match (Internet from 23-Sep-2008)
http://www.ime.usp.br/~egbirgin/publications/bmrsurveyspg.pdf

79 < 1% match (publications)
Koos, S., J. Mouret, and S. Doncieux. "The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics", IEEE Transactions on Evolutionary Computation, 2012.

80 < 1% match (publications)
Francisco J. Valero-Cuevas. "", IEEE Transactions on Biomedical Engineering, 11/2007

81 < 1% match (student papers from 19-Feb-2012)
Submitted to King Saud University on 2012-02-19

82 < 1% match (student papers from 30-Nov-2003)
Submitted to Boston University on 2003-11-30

83 < 1% match (student papers from 10-Jan-2013)
Submitted to King Fahd University for Petroleum and Minerals on 2013-01-10

84 < 1% match (student papers from 05-Mar-2014)
Submitted to University of Newcastle upon Tyne on 2014-03-05

85 < 1% match (student papers from 19-Jun-2015)
Submitted to 87988 on 2015-06-19

86 < 1% match (student papers from 04-Dec-2013)
Submitted to Cranfield University on 2013-12-04

87 < 1% match (student papers from 25-May-2015)
Submitted to Imperial College of Science, Technology and Medicine on 2015-05-25

88 < 1% match (student papers from 13-Sep-2013)
Submitted to University of Newcastle upon Tyne on 2013-09-13

89 < 1% match (student papers from 30-Apr-2015)
Submitted to University Tun Hussein Onn Malaysia on 2015-04-30

90 < 1% match (student papers from 01-Nov-2012)
Submitted to Rochester Institute of Technology on 2012-11-01

91 < 1% match (student papers from 10-Dec-2012)
Submitted to Oklahoma State University on 2012-12-10

92 < 1% match (Internet from 16-Mar-2008)
http://www.ri.cmu.edu/people/dellaert_frank_text.html

93 < 1% match (Internet from 30-Aug-2014)
http://flyingmachinearena.org/wp-content/publications/2013/ritIROS13.pdf

94 < 1% match (student papers from 17-May-2013)
Submitted to University of Southampton on 2013-05-17

95 < 1% match (Internet from 03-Jun-2014)
http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2012-014r002.pdf

96 < 1% match (Internet from 19-Mar-2010)
http://www.cs.cmu.edu/afs/cs/misc/mosaic/common/omega/Web/People/gjb/includes/publications/journal/nelson2009-ras/nelson2009-ras.pdf

97 < 1% match (Internet from 05-Nov-2012)
http://www.reginafrei.ch/pdf/future-of-CE-published.pdf

98 < 1% match (Internet from 28-Sep-2010)
http://www.mat.univie.ac.at/~neum/glopt/mss/SchRF03.pdf

99 < 1% match (Internet from 21-Nov-2008)
http://www3.lehigh.edu/images/userImages/jgs2/Page_7287/LU-CSE-07-004.pdf

100 < 1% match (Internet from 11-Jan-2014)
http://ijcas.com/admin/paper/files/IJCAS_v2_n4_pp463-474.pdf

101 < 1% match (Internet from 19-Aug-2014)
http://meloncholy.com/wordpress/wp-content/uploads/thesis-andrew-weeks.pdf

102 < 1% match (Internet from 24-Oct-2011)
http://webpages.iust.ac.ir/yaghini/Courses/AOR_872/Ant%20Colony%20Optimization_01.pdf

103 < 1% match (publications)
GUO-YUAN QIU and SHIH-HUNG WU. "The Evolutionary Locomotion of Tripedal and Quadrupedal Biomorphic Robots", Journal of Information Science & Engineering, 2013.

104 < 1% match (student papers from 05-Mar-2013)
Submitted to University of Sheffield on 2013-03-05

105 < 1% match (student papers from 05-Dec-2011)
Submitted to University of Oklahoma on 2011-12-05

106 < 1% match (student papers from 25-May-2015)
Submitted to University of Sheffield on 2015-05-25

107 < 1% match (student papers from 22-Jun-2012)
Submitted to Universiti Teknologi Malaysia on 2012-06-22

108 < 1% match (Internet from 03-Nov-2010)
http://lsro.epfl.ch/page69906-en.html

109 < 1% match (Internet from 28-Sep-2010)
http://evelyne.lutton.free.fr/Papers/RR-6914.pdf

110 < 1% match ()
http://pakhawaj.cs.uiuc.edu/~srinivsn/Pubs/ruchira2.pdf

111 < 1% match (Internet from 11-Jun-2011)
http://www.ini.rub.de/thbio/members/profil/Rano/index_p.html

112 < 1% match (publications)
Mirmomeni, Masoud, and William Punch. "Co-evolving data driven models and test data sets with the application to forecast chaotic time series", 2011 IEEE Congress of Evolutionary Computation (CEC), 2011.

113 < 1% match (student papers from 31-Aug-2015)
Submitted to University of Sheffield on 2015-08-31

114 < 1% match (student papers from 06-Jul-2015)
Submitted to International Islamic University Malaysia on 2015-07-06

115 < 1% match (student papers from 28-Aug-2014)
Submitted to University of Queensland on 2014-08-28

116 < 1% match (Internet from 06-Jun-2012)
http://kth.diva-portal.org/smash/get/diva2:12636/FULLTEXT01

117 < 1% match (Internet from 25-Aug-2014)
http://www.tech-uofm.info/fall_2013/TECH3232/TI_Digital_Logic_Pocket_Databook.pdf

118 < 1% match (Internet from 12-Oct-2014)
http://cowsandfish.org/publications/documents/GreencoverRiparianSoilsSABReportbyLandWise.pdf

119 < 1% match (Internet from 15-Sep-2014)
http://www.macdorman.com/kfm/writings/pubs/MacDormanCowley2006LTRbenchmark.pdf

120 < 1% match (Internet from 04-Dec-2013)
http://cecs.louisville.edu/ry/AIcompleteZero.pdf

121 < 1% match (Internet from 19-Jun-2011)
http://www-2.cs.cmu.edu/~biglou/captcha.pdf

122 < 1% match (Internet from 21-Sep-2011)
http://www.ceng.metu.edu.tr/research/kovan/index

123 < 1% match ()
http://www.sens.ee.saga-u.ac.jp/wakuya/paper/nc-03-03a.pdf

124 < 1% match (Internet from 03-Dec-2013)
http://www.fh-meschede.de/public/willms/research/willms_RunStructureTIT2259293.pdf

125 < 1% match (Internet from 22-Aug-2011)
http://stat.asu.edu/~platte/pub/RBFconvergenceIMANA_REV2.pdf

126 < 1% match (student papers from 09-May-2011)
Submitted to University of Sheffield on 2011-05-09

127 < 1% match (student papers from 06-Jun-2014)
Submitted to Imperial College of Science, Technology and Medicine on 2014-06-06

128 < 1% match (Internet from 31-Aug-2010)
http://heikohamann.de/pub/hamannCEC10.pdf

129 < 1% match (Internet from 07-Nov-2008)
http://www.cs.bgu.ac.il/~sipper/papabs/gpgames.pdf

130 < 1% match (Internet from 15-Dec-2012)
http://www.ntspress.com/wp-content/uploads/2012/07/Eureka-Part-2.pdf

131 < 1% match (Internet from 22-Sep-2010)
http://evelyne.lutton.free.fr/Papers/SapinEmmanuel_LouchetJean_LuttonEvelyne_05-07-2009_paper.pdf

132 < 1% match (Internet from 15-Nov-2010)
http://www.fl.ctrl.titech.ac.jp/member/ibuki/paper/IHFS_CDC10.pdf

133 < 1% match (Internet from 23-Jan-2014)
http://www.cs.unc.edu/cms/publications/dissertations/galoppo.pdf/at_download/file

134 < 1% match (Internet from 26-Aug-2014)
http://www.clips.uantwerpen.be/sites/default/files/modeling-creativity.pdf

135 < 1% match (Internet from 24-Oct-2014)
http://stumptown.cc.gt.atl.ga.us/cse6230-hpcta-fa11/slides/25-cnc.pdf

136 < 1% match ()
http://www.recherche.enac.fr/opti/facile/doc/libref/index_values.html

137 < 1% match (publications)
Wu, Min-Thai Hong, Tzung-Pei Lee, Chung-. "Using the ACS approach to solve continuous mathematical problems in engineering.(Research Article)(a", Mathematical Problems in Engineering, Annual 2014 Issue

138 < 1% match (student papers from 10-Aug-2014)
Submitted to National University of Singapore on 2014-08-10

139 < 1% match (Internet from 13-Sep-2014)
http://www.metaheuristics.net/~mdorigo/HomePageDorigo/publicationsbytype.php

140 < 1% match (Internet from 27-Jan-2014)
http://miguelduarte.pt/media/Miguel_Duarte_MSc_Thesis.pdf

141 < 1% match (Internet from 21-Oct-2010)
http://web.cs.msu.edu/~dk/papers/knoester2010neuroevolution.pdf

142 < 1% match (Internet from 23-Jul-2008)
http://www.nclab.tw/index.php?option=com_docman&task=doc_download&gid=62&Itemid=39

143 < 1% match (Internet from 28-Oct-2010)
http://www.swarmanoid.org/upload/pdf/Ampatzis-EtAl:ECAL:09.pdf

144 < 1% match (Internet from 01-Oct-2014)
http://infoscience.epfl.ch/record/177201/files/biorob_2012.pdf

145 < 1% match (Internet from 09-Jun-2012)
http://mavrinac.com/files/academic/mavrinac11_3drec.pdf

146 < 1% match (Internet from 21-Feb-2012)
http://www.cs.iusb.edu/~danav/papers/dv_eit07.pdf

147 < 1% match ()
http://www.ee.siue.edu/~sumbaug/439robotimages/sauser8.tif*

148 < 1% match (Internet from 12-Jun-2012)
http://www.kidresearch.jp/publications/thesis-Nakakoji93.pdf

149 < 1% match (Internet from 19-Nov-2012)
http://www.init.ituniv.se/~palle/publications/SoundsUnh-pap6.pdf

150 < 1% match (Internet from 09-Oct-2011)
http://www.cs.le.ac.uk/people/nwalkinshaw/Files/icgi08.pdf

151 < 1% match (Internet from 15-Sep-2014)
http://iesashift.nl/wp-content/uploads/group-documents/5/1288088732-Bollinger-GR.pdf

152 < 1% match (Internet from 07-Sep-2014)
http://edoc.hu-berlin.de/dissertationen/bossan-benjamin-2013-10-21/PDF/bossan.pdf

153 < 1% match (Internet from 08-Oct-2014)

http://en.wikipedia.org/wiki/Swarm_behaviour

154 < 1% match (Internet from 18-Oct-2010)
http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2005-024r001.pdf

155 < 1% match (Internet from 07-Feb-2014)
http://edoc.ub.uni-muenchen.de/16071/1/Seiler_Christian.pdf

156 < 1% match (Internet from 15-Apr-2015)
http://ssg.mit.edu/ssg_theses/ssg_theses_1974_1999/Chou_PhD_5_91.pdf

157 < 1% match (Internet from 05-Dec-2010)
http://www.academypublisher.com/jnw/vol03/no07/jnw03072633.pdf

158 < 1% match (publications)
"Findings from J.R. Tumbleston and Co-Authors Broaden Understanding of Science (Additive manufacturin", Science Letter, April 17 2015 Issue

159 < 1% match (student papers from 22-Apr-2014)
Submitted to Flinders University on 2014-04-22

160 < 1% match (Internet from 14-Oct-2014)
http://www.crcsi.com.au.stage.gs/getattachment/6b3f4dd2-2784-4a38-ab41-d3536baba6af/Robust-and-Efficient-Hardware-based-Evolutionary-T.aspx

161 < 1% match (Internet from 10-May-2011)
http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2006-022r002.pdf

162 < 1% match (Internet from 01-Nov-2014)
http://spinal.evsc.net/media/Thesis_Spinal_Rhythms.pdf

163 < 1% match (Internet from 14-Oct-2010)
http://ir.canterbury.ac.nz/bitstream/10092/4620/1/thesis_fulltext.pdf

164 < 1% match (Internet from 25-Oct-2011)
http://www.ri.cmu.edu/pub_files/pub4/kalra_nidhi_2005_1/kalra_nidhi_2005_1.pdf

165 < 1% match (Internet from 23-Aug-2014)
http://homepages.dcc.ufmg.br/~chaimo/public/IROS09b.pdf

166 < 1% match (Internet from 24-Aug-2014)
http://130.217.226.8/bitstream/handle/10652/2031/Lei%20Zhu_2012-09-14.pdf?sequence=1

167 < 1% match (Internet from 21-Jan-2011)
http://robotics.dei.unipd.it/~fabiodl/pdfs/interleaving.pdf

168 < 1% match (Internet from 08-Feb-2014)
http://www.mtome.com/Publications/JMLR/jmlr-vol7-partB.pdf

169 < 1% match (Internet from 23-Aug-2014)
http://www.ic.unicamp.br/~reltech/2011/11-15.pdf

170 < 1% match (Internet from 22-Jan-2014)
http://150.214.190.154/docencia/sf1/Rahnamayan08.pdf

< 1% match (Internet from 11-Jun-2008)

171 http://www.tu-dresden.de/etifwt/wwwroot/mitarbeiter/lienig/trans_ec.pdf

172 < 1% match (Internet from 05-Jun-2008)
http://cs.gmu.edu/~lpanait/papers/panait03improving.pdf

173 < 1% match (Internet from 27-Nov-2008)
http://www.ir.bbn.com/~vkawadia/papers/diss.pdf

174 < 1% match (Internet from 21-Jun-2011)
http://www.isi.edu/~lerman/papers/lerman-alife.pdf

175 < 1% match (Internet from 07-Oct-2010)
http://ima.ac.uk/papers/baxter2007.pdf

176 < 1% match (Internet from 16-Sep-2014)
http://run.unl.pt/bitstream/10362/10227/1/Santos_2011.pdf

177 < 1% match (Internet from 16-Aug-2014)
http://ec.europa.eu/economy_finance/publications/publication12832_en.pdf

178 < 1% match (Internet from 28-Nov-2013)

http://ir.sia.ac.cn/bitstream/173321/10587/2/RFID%20network%20scheduling%20using%20a%20discrete%20multi-swarm%20optimizer.pdf

179 < 1% match (Internet from 10-Oct-2010)
http://cs.brynmawr.edu/~dkumar/Myro/Text/June09/PDF/Chapter6.pdf

180 < 1% match (Internet from 31-Dec-2010)
http://hal.eeng.nuim.ie/~jringwood/Respubs/J152TSM.pdf

181 < 1% match (Internet from 21-Feb-2015)
http://helvia.uco.es/xmlui/bitstream/handle/10396/10945/2013000000822.pdf.txt?sequence=3

182 < 1% match (Internet from 20-Feb-2015)
http://www.cond-mat.de/events/correl14/manuscripts/correl14.pdf

183 < 1% match (Internet from 05-Oct-2011)

http://www.ifd.dauphine.fr/fileadmin/mediatheque/recherche_et_valo/FDD/CAHIER21_EKELAND_ET_AL.pdf

184 < 1% match (Internet from 17-Oct-2008)
http://nicosia.is.s.u-tokyo.ac.jp/mp/hagiya-final.pdf

185 < 1% match (Internet from 27-Dec-2012)
http://www.cs.mcgill.ca/~chang/pub/gpsrobust.pdf

186 < 1% match (publications)
"Swarm Intelligence in Optimization and Robotics", Springer Handbook of Computational Intelligence, 2015.

187 < 1% match (publications)
Stéphane Doncieux. "Evolutionary Robotics: Exploring New Horizons", Studies in Computational Intelligence, 2011

188 < 1% match (publications)

BAYINDIR, Levent and ŞAHİN, Erol. "A review of studies in swarm robotics", TÜBİTAK, 2007.

---

189 < 1% match (publications)

ZAFAR, Kashif and BAIG, Abdul Rauf. "MULTIPLE ROUTE GENERATION USING SIMULATED NICHE BASED PARTICLE SWARM OPTIMIZATION", Computing & Informatics, 2013.

---

190 < 1% match (publications)

Kenneth De Jong. "The effects of interaction frequency on the optimization performance of cooperative coevolution", Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO 06 GECCO 06, 2006

---

191 < 1% match (publications)

Journal of Modelling in Management, Volume 1, Issue 3 (2007-07-22)

---

192 < 1% match (publications)

Cortes Martinez, Facundo Cansino, Alejan. "Mathematical analysis for the optimization of a design in a facultative pond: indicator organism and", Mathematical Problems in Engineering, Annual 2014 Issue

---

193 < 1% match (student papers from 01-Jun-2014)

Submitted to University of New South Wales on 2014-06-01

---

194 < 1% match (publications)

Nelson, A.L.. "Fitness functions in evolutionary robotics: A survey and analysis", Robotics and Autonomous Systems, 20090430

---

195 < 1% match (publications)

Scheuermann, Bernd. "Ant colony optimization on runtime reconfigurable architectures", Universität Karlsruhe, 2005.

---

196 < 1% match (student papers from 25-Nov-2011)

Submitted to Bolton Institute of Higher Education on 2011-11-25

---

197 < 1% match (Internet from 29-Nov-2013)

http://eprints.uwe.ac.uk/12746/1/TAROS2010_PaulODowd.pdf

---

198 < 1% match (Internet from 20-Apr-2012)

http://www.replicators.eu/BV/symbrion/tiki-index.php?page=Publications

---

199 < 1% match (publications)

Kernbach, . "Bibliography", Handbook of Collective Robotics Fundamentals and Challenges, 2013.

---

200 < 1% match (publications)

Kopman, Vladislav, Giovanni Polverino, Jeffrey Laut, and Maurizio Porfiri. "Using a Bioinspired Robotic-Fish for Closed-Loop Control of Zebrafish Response in a Preference Test", Volume 2 Legged Locomotion Mechatronic Systems Mechatronics Mechatronics for Aquatic Environments MEMS Control Model Predictive Control Modeling and Model-Based Control of Advanced IC Engines, 2012.

---

201 < 1% match (publications)

Nelson, A.L.. "Using direct competition to select for competent controllers in evolutionary robotics", Robotics and Autonomous Systems, 20061031

---

202 < 1% match (publications)

Doncieux, Stephane, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E. (Gusz) Eiben. "Evolutionary Robotics: What, Why, and Where to", Frontiers in Robotics and AI, 2015.

---

203 < 1% match (publications)
Koos, Sylvain, Antoine Cully, and Jean-Baptiste Mouret. "High resilience in robotics with a multi-objective evolutionary algorithm", Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion - GECCO 13 Companion, 2013.

204 < 1% match (publications)
"Three-fold Adaptivity in Groups of Robots : The Effect of Social Learning", Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO 15, 2015.

205 < 1% match (publications)
Greenspoon, S.A. Sykes, K.L.V. Ban, J.D., "Automated PCR setup for forensic casework samples using the Normalization Wizard and PCR Setup robot", Forensic Science International, Dec 20 2006 Issue

206 < 1% match (student papers from 11-Aug-2014)
Submitted to University of Edinburgh on 2014-08-11

207 < 1% match (student papers from 13-Mar-2014)
Submitted to University of Leeds on 2014-03-13

208 < 1% match (Internet from 21-Nov-2008)
http://iridia0.ulb.ac.be/IridiaTrSeries/IridiaTr2006-023r001.pdf

209 < 1% match (student papers from 31-May-2015)
Submitted to University of Northampton on 2015-05-31

210 < 1% match (Internet from 11-Jan-2008)
http://jeb.biologists.org/cgi/content/full/208/5/915

paper text:

Automated Reverse Engineering of Agent Behaviors Wei Li

11**A thesis** submitted **for the degree of Doctor of Philosophy Department of Automatic Control and Systems Engineering The University of Sheffield**

30th September 2015 Abstract This thesis concerns the automated reverse engineering of agent behaviors. It proposes a metric-free coevolutionary approach—Turing Learning, which

7**allows a machine to infer the** behaviors **of**

agents (simulated or physical ones), in a fully automated way. Turing Learning consists of two populations. A

4**population of models competitively coevolves with a population of classifiers. The classifiers observe the models and agents. The fitness of the classifiers depends solely on their ability to** distinguish **between them. The** models, on **the**

other hand,

33**are evolved to mimic the behavior of** the agents **and**

mislead the judgment of the classifiers. The

4fitness of the models depends solely on their ability to 'trick' the classifiers into categorizing them as agents. Unlike other methods for system identification, Turing Learning does not require any predefined metrics to quantitatively measure the difference between the models and agents. The merits of Turing Learning

are demonstrated using three

10case studies. In the first case study, a

machine automatically infers the behavioral rules of a group of homogeneous agents through observation.

1A replica, which resembles the agents under investigation in terms of behavioral capabilities, is

mixed into the group. The models are executed on the replica. This case study is conducted with swarms of both simulated and physical robots. In the second and third case studies, Turing Learning is applied to infer deterministic and stochastic behaviors of a single agent through controlled interaction, respectively. In particular, the machine is able to modify the environmental stimuli that the agent responds to. In the case study of inferring deterministic behavior, the machine can construct static patterns of stimuli that facilitate the learning process. In the case study of inferring stochastic behavior, the machine needs to interact with the agent on the fly through dynamically changing patterns of stimuli. This allows the machine to explore the agent's hidden information and thus reveal its entire behavioral repertoire. This interactive approach proves superior to learning only through observation. iii Acknowledgments I consider the process of pursuing my PhD as a journey to learn not only about sci- ence, but also about life. Firstly,

30I would like to thank my supervisors, Dr. Roderich Groß and

Prof. Stephen A. Billings for their support in this journey. The special thanks would be given to Roderich. He helped me from toddling to walking on the pathway in this research field by motivating, teaching and passing his professional experience to me without any reservation. He is always very patient to discuss with me and give me professional suggestions. His passion for pursuing science and rigorous attitude in research have significantly influenced me. Second, many people have contributed to this journey, especially people in the Natural Robotics Lab: Melvin Gauci, Jianing Chen, Yuri K. Lopes, Christopher Parrott, Fer- nando Perez Diaz, Stefan Trenkwalder, Gabriel Kapellmann Zafra, Matthew Doyle, and Shen-Chiang Chen, who make the lab a warm, supportive and fun environment. Special thanks to Melvin Gauci and Jianing Chen. To Melvin, I considered myself very lucky that I met him from the start of this unforgettable journey. I have learned much from him on doing research and life. We not only had many collaborations on research, but we also shared the experience outside science. To Jianing, I appreciate his help for sharing his knowledge with me and providing valuable feedback when I did my experiments. Outside the research group, there are also a number of people who helped me during my stay in Sheffield, especially Xiao Chen and Yuzhu Guo. This journey would not be completed without their encouragement.

151Finally, I would like to thank my family. To my parents, thank you for

giving me a free environment to grow up and always supporting me without any hesitation. To my wife, Yifei, thank you for always accompanying and encouraging me, which makes me maintain a positive attitude for life. v Contents Abstract Acknowledgments

. . . . . .

131 **6.2 Future work**

. . . . . 132 x 1 Intro duction 1.1 Motivation and Objectives

3**Over the last 50 years,** robotic **and automation** systems **have** transformed **our world**

and greatly enhanced the quality of our daily life.

3**With the development of science and technology,** many **intelligent**

systems that

3**integrate machines, electronics, control and information technologies have emerged. Such systems can accomplish numerous tasks originally performed by humans and often prove superior in terms of precision, speed and cost. They can replace humans in the tasks that require repetitive and monotonous operations.**

For example, in the automotive industry, robotic and automation systems have been widely used for machining, welding, painting and assembling. From an in- dustrial perspective, these systems have lowered the cost and improved the efficiency of production, thus greatly increasing the speed of industrialization. Robotic and automation systems also contribute to scientific research. In outdoor envi- ronments such as those that may be beyond humans' capabilities of reach, robots can be sent to collect samples or conduct experiments. For example, two robots—Spirit and Opportunity were sent to Mars in 2004 by NASA in a mission to explore the geology of this planet [1]. The primary goal of this mission was to analyze the rocks and soils on Mars and explore the activity of water in the past. In indoor laboratories, robotic and automation systems have also played an important role.

3**With the** help **of** these **systems, researchers can collect data much faster than ever before.**

For instance, high- throughput screening (HTS) systems [2], which are widely used in drug discovery, allow the researchers to conduct millions of experiments in a very short time. Such systems consist of several components, including sensing units, robotic manipulators, control sys- tems, etc. Besides data collection, robotic and automation systems can also help analyze 1 1 Introduction the data

3**using intelligent software, which provides an** effective **tool for data analysis in scientific research and frees** researchers **from the tedious and** monotonous **process of data analysis if done manually.**

This accelerates the development of scientific research to a great extent. A particular scientific area in which robotic and automation systems play a significant role is ethology—the study of animal behavior [3].

3**There are four types of questions to be investigated in**

ethology:

3**questions concerning causes, functions, development and evolution** [3]. **Causes refer to the mechanisms of animals that are innate as well as the external/** internal **stimuli that affect such** behavior. **Functions concern what is**

the purpose of this

behavior such as mating, aggregation or foraging. The development of animal behavior concerns

3**how animals learn such** behavior **during their life** and **how such** behavior **is affected by experience, while evolution relates to how the** behavior **changes over generations in the course of natural evolution.**

Over centuries, these four questions have been investigated by ethologists primarily in well-controlled (indoor) laboratories or outdoor environments. Ethology

1**is pursued not only because it is a subject of interest in itself, but also because the knowledge gained from it has**

1**several practical applications. For instance, models of** animal **decision-making can be used to predict their behavior in novel environments, which can help in** making **ecological conservation policy** [4]. **Knowledge about animal** behavior **has also been applied for solving** computational **problems** [5], **and for constructing biologically-inspired robotic agents**

[6]. Before the emergence of computers, ethologists needed to observe the animals and an- alyze the data manually. They also needed to learn

2**how to control the environmental conditions in a meaningful way**

to extract most of the information from the animals under investigation. Such process of analysis sometimes is time-consuming and tedious. With the help of intelligent and automation systems, nowadays researchers can conduct experiments much more efficiently. However, these systems are often secondary, and in most of the cases they are merely accomplishing mechanical and repetitive work. The question is whether we can build a machine/system that can accomplish the whole pro- cess of scientific investigation and automatically analyze experimental data, search for correlations between different elements, generate new hypotheses and devise new ex- perimental conditions to be investigated. In

134**other words, can we** build **a system that is able to**

automatically conduct scientific research without (or with minimal) human 2 1.1 Motivation and Objectives intervention? Recently, the development of “robot scientists” shows that such systems may be within reach [7, 8, 9]. Following this motivation, this thesis aims to pave the way for further development in science automation [8], especially in the area of ethology. The ultimate goal of this thesis is to contribute to the study of animal behavior through developing an automated system identification method. System identification is a process of modeling natural or artificial systems with observed data. It has drawn a large interest among researchers for decades [10, 11]. One appli- cation of system identification is the reverse engineering of agent behavior (biological organisms or artificial agents). Many studies have investigated how to deduce rules of agent behavior using system identification techniques based on various models [12], one of which is an agent-based model [13], which simulates the complex behavior of a group of agents with relatively simple behavioral rules. The global behavior emerging from interaction within agents and between agents and environments is used for refining the model. Evolutionary computation which draws inspiration from biological evolution has proven to be a powerful method to automate the generation of models, especially for behaviors that are hard to formulate [14, 15, 16]. Evolutionary computation also pro- vides a potential realization for automation science, as models evolve in an autonomous manner. It is the main technique that is investigated in this thesis for performing system identification. A limitation of existing system identification methods is that they rely on predefined metrics, such as the sum of squared error, to measure the difference between the output of the models and that of the system under investigation. Model optimization then proceeds by minimizing the measured differences. However, for complex systems, defining a metric can be

non-trivial and case-dependent. It may require significant prior information about the systems. Moreover, an unsuitable metric may not well distinguish between good and bad models, or even bias the identification process. This thesis aims to overcome the limitation by introducing a metric-free system identification

> 2**method that allows a machine to infer** agent **behavior**

automatically. 1.2 Problem Statement The investigated (agent) behaviors in this thesis are simulated by a computer or physical robots. The agent to be studied is put in an environment. Its behavior depends on interaction with the environment and with other agents in a group (if any). The machine will observe the agent's motion. It is assumed

> 2**that it is possible to track the position** and orientation **of the** agent **at discrete steps in time.**

In general, one could monitor a range of variables including the agent's motion, heart rate, morphology, body temperature, etc. Furthermore, the machine can control environmental stimuli that the agent may respond to. The system identification task is to infer the observed behavior, in other words, the agent's behavioral rules. Three

> 176**case studies** are **presented in this thesis. The first**

case study is about infer- ring

> 1**swarm behaviors,** which **are emergent behaviors that arise from the interactions of**

numerous simple individuals [17]. Learning about behaviors that are exhibited in a collective manner

> 7**is particularly challenging, as the individuals not**

only interact with the environment but also with each other. Typically their motion appears stochastic and is difficult to predict [18]. For instance, given a swarm of simulated fish, one would have to evaluate how close its behavior is to that of a real fish swarm, or how close the individual behavior of a simulated fish is to that of a real fish. Comparing behaviors at the level of the swarm (i.e., emergent behaviors) is challenging [19]. It may require domain-specific knowledge and not discriminate among alternative individual rules that exhibit similar collective dynamics [20]. Comparing behaviors at the level of individuals is also difficult, as even the same individual fish in the swarm is likely to exhibit a fun- damentally different trajectory every time it is being looked at. In this case study, the machine observes the motion of each individual in the swarm and needs to automatically infer the behavioral rules of the swarming agents. The second case study is about inferring the deterministic behaviors of a single agent, and investigating how the agent responds to the environmental stimuli. The machine has full control over the environmental stimuli that the agent may respond to, and at the same time observes the agent's motion. We investigate a particular type of agent behavior; from the perspective of system identification, the agent behavior has low observability. That is, randomly generated sequences of inputs (stimuli) may not be sufficient to extract 4 1.3 Contributions all the hidden information of the agent and thus infer its behavior. Instead, in order to reveal all the agent's behavioral repertoire, the machine needs to construct complex patterns of stimuli in particular ways that facilitate the learning process. In the third case study, we investigate how to infer the stochastic behaviors of a single agent. The agent still responds to the environmental stimuli; however, its behavior is not only determined by the environmental stimuli but also by probability. In other words, constructing a fixed sequence of stimuli may not trigger all the agent's behavioral reper- toire as mentioned in the second case study. To reveal the agent's hidden behavior, the machine needs to interact with the agent during the experimental process and dynam- ically change/control the environmental stimuli based on the agent's observed motion. Inferring such stochastic behaviors is challenging, as given the same sequence of stimuli, the agent is likely to exhibit different behaviors. 1.3 Contributions The main contribution of this thesis is a novel system identification approach—Turing Learning—which

9**allows a machine to infer** agent **behavior** in **an**

autonomous manner. Turing Learning

9**uses a coevolutionary algorithm** comprising **two populations.**

A popu- lation of

4**models competitively coevolves with a population of classifiers. The classifiers observe the models and agents. The fitness of the classifiers depends solely on their** abil- ity **to discriminate between them. Conversely, the fitness of the models depends solely on their ability to 'trick' the classifiers into categorizing them as agents. Unlike other** system identification **methods, Turing Learning does not** rely on **predefined metrics to** gauge **the difference between** the behaviors of **agents**

and models. The specific contributions are as follows: 1) A metric-free approach to automatically infer the behavioral rules of a group of ho- mogeneous agents. The approach was first validated in simulation; both the inferred model parameters and emergent global behaviors closely matched those of the original system. 2) A metric-free approach to automatically produce a classifier (system) that can be used for detecting abnormal behaviors (e.g., faulty agents in a swarm). This was validated by both simulated and physical robot swarms. 3) A physical system for performing metric-free system identification in an autonomous manner. The system was validated through successfully inferring the behavioral rules of a physical robot swarm. Both the inferred model parameters and emergent global behaviors closely matched those of the original system. 4) A metric-free approach to automatically infer deterministic behavior of a single agent. The machine actively constructs complex patterns of inputs (stimuli) to extract the agent's entire behavioral repertoire. 5) A metric-free approach to automatically infer stochastic behavior of a single agent through controlled interaction. The machine changes on the fly an environmental stimulus depending on the output of the agent under observation. 1.4 Publications This thesis presents the author's own work. Some

16**parts of the thesis have been published. A**

preliminary version of Chapter 3 was orally presented in a conference by the author of the thesis: •

5**W. Li, M. Gauci and R. Groß,**"Coevolutionary learning of swarm behaviors with- out metrics," **Proceedings of 2014 Genetic and Evolutionary Computation Conference (GECCO** 2014). **ACM Press,**

Vancouver, Canada, 2014, pp. 201–208. A preliminary version of Chapter 5 was orally presented in a conference by the author of this thesis: •

5**W. Li, M. Gauci and R. Groß,**"A coevolutionary approach to learn animal behav- ior through controlled interaction," **Proceedings of 2013 Genetic and Evolutionary Computation Conference (GECCO** 2013). **ACM Press,**

Amsterdam, The Nether- lands, 2013, pp. 223–230. 6 1.4 Publications Preliminary versions of Chapters 3 and 4 have been included in a paper submitted to the following journal: •

7**W. Li, M. Gauci,** J. Chen **and R. Groß,**

"Reverse engineering swarm behaviors through turing learning,"

7IEEE Transactions on Evolutionary Computation,

under review. Apart from the work presented in this thesis, the author has also contributed to other projects. This led to the following publications: •

5M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß,"Self-organized aggregation without computation," The International Journal of Robotics Research, vol. 33, no. 8, pp. 1145–1161, 2014.• J. Chen, M. Gauci,

18W. Li, A. Kolling and R. Groß, "Occlusion-based coopera- tive transport with a swarm of miniature mobile robots." IEEE Transactions on Robotics, vol. 31, no. 2, pp. 307–321, 2015.

•

1M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß,"Clustering objects with robots that do not compute," in Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems

(AAMAS 2014). IFAAMAS Press, Paris, France, 2014, pp. 421–428. During his PhD studies, the author has also been a Marie Curie Research Fellow

145with the Department of Mechanical Engineering, University of Western Ontario, Canada,

where he contributed to a project on Mechanical Cognitivization. This led to the following publication: • G. Avigad, W. Li, A. Weiss, "Mechanical Cognitivization: A kinematic system proof of concept" Adaptive Behavior, vol. 23, no. 3, pp. 155–170, 2015. 1.5

35Thesis Outline This thesis is structured as follows: • Chapter 2 describes the background of the

thesis as well as the related work pre- sented in the literature. • Chapter 3 introduces a metric-free system identification method—Turing Learn- ing. It is applied to learn

1two swarm behaviors (self-organized aggregation [21] and self-organized object

clustering [22]) through observation. Section 3.1 describes the implementation of Turing Learning and the two swarm behaviors. Section 3.2 introduces the simulation platform and setups for performing coevolution runs. Section 3.3 presents the results obtained from the two case studies. In particular, Section 3.3.1 analyzes the evolution of models, objectively measuring their quality in terms of local and global behaviors. Section 3.3.2 investigates the coevolution- ary dynamics. Section 3.3.3 investigates the evolution of classifiers, showing how to construct a robust classifier system to potentially detect abnormal behaviors in the swarm. Section 3.3.4 studies the effect of observing only a subset of agents in the swarm. Section 3.3.5 presents a study where an aspect of the agents' mor- phology (their field of view) and brain (controller) are inferred simultaneously. Section 3.3.6 shows the results of using Turing Learning to learn other swarm be- haviors. Section 3.3.7 presents a study showing the method's sensitivity to noise. Section 3.4 summaries the findings in this chapter. • Chapter 4 presents a real-world validation of Turing Learning to infer the behav- ior of a swarm of physical robots. Section 4.1 introduces the physical platform, which includes the robot arena, the robot platform and the sensors implementa- tion. Section 4.2 details the tracking system, including

motion capture and video processing. Section 4.3 describes the programs executed by each component (ma- chine, agent and replica) during the coevolutionary learning process. Section 4.4 describes the experimental setup. Section 4.5 discusses the results obtained. This includes the analysis of the evolved models and the analysis of the evolved classi- fiers. Section 4.6 analyzes the sensitivity of Turing Learning for individual failure during the experimental trial. Section 4.7 summaries the results obtained and discusses the findings in this chapter. 1.5 Thesis Outline • Chapter 5 investigates how to infer the deterministic and stochastic behaviors of an agent through interactive Turing Learning. The machine not only observes the behavior of the agent but also interacts with it through changing the stimulus that influences the agent's behavior.

> 155**This chapter is** organized **as follows. Sec- tion** 5 **.2** describes **the**

methodology, illustrating how Turing Learning is extended to have interactive capability. The deterministic and stochastic behaviors under investigation are presented as two case studies (Sections 5.3 and 5.4). Section 5.3.1 describes the deterministic behavior. Section 5.3.2 presents the simulation setup.

> 30**Section 5.3.** 3 **presents the results of** inferring **the**

deterministic behavior. Sec- tion 5.4.1 describes the stochastic behavior (using a state machine) for the general case. Section 5.4.2 presents the simulation setup for inferring the stochastic be- havior. Sections 5.4.3 and 5.4.4 present the obtained results for the case of 2 states and 3 states, respectively. Section 5.5 summaries the chapter. • Chapter 6

> 90**concludes the thesis and discusses the future work. 2 Background and Related Work This chapter**

comprises three parts. Section 2.1 briefly introduces the background, in- cluding the development of artificial intelligence (AI) and robotics, and some related work in automation science. Section 2.2 reviews the development of evolutionary com- putation, which is the main technique used in this thesis. It introduces biological evo- lution, the principles, strengths and weaknesses of evolutionary computation, and its applications. Section 2.3 reviews how AI/robotics and animal behavior study benefit from each other. It gives examples of how animal behavior inspires AI/robotics and of how to investigate animal behavior using AI/robotics techniques. 2.1 Background 2.1.1 Development of AI and Robotics Intelligence is a natural part of life. Humans and other biological creatures exhibit many behaviors (e.g., decision making and foraging) that require intelligence. However, intel- ligence

> 3**is not a property that** is **limited to biological creatures. It should be equally applicable to machines.** The term artificial **intelligence**

emerged in a conference in 1956 at Dartmouth College, where several researchers (e.g., Marvin Minsky and John Mc- Carthy) who were later considered pioneers of this field, discussed the development of digital computers and the future of AI. The definition of AI is still a disputed topic. Some researchers argue that AI

> 3**is to simulate the intelligent** behaviors that **are observed in humans and other** biological **creatures using computers or machines.**

That is, an intelligent machine should be able to exhibit behaviors similar to that of living crea- tures when encountering the same problems [23]. Others give the following definition: 2 Background and Related Work "Artificial Intelligence is the study of mental faculties through the use of computational models" [24]. According to Fogel [25], an intelligent system should know how to make decisions in order to fulfill a goal (e.g., solving a problem). In other words, instead of pre-programming the machine using human's knowledge, it should be able to learn and adapt. In [26], Minsky argued, "Why can't we build, once and for all, machines that grow and improve themselves by learning from experience? Why can't we simply explain what we want, and then let our machines do experiments or read some books or go to school, the sorts of things that people do?" In 1950, Turing [27] proposed an

163**imitation game,** which **is** nowadays **known as the Turing test,**

to discuss a question: “Can machine think?”. Although whether a machine could pass the Turing test or not was beyond the consideration at that time, it was accepted as a notion that a machine could learn and adapt to mimic human behavior. In the 1970s, the emergence of expert systems—computer programs that mimic human experts’ decision-making capability [28], significantly promoted the development of AI. An expert system can solve complicated problems through reasoning about the knowl- edge (which is mainly represented as if-then rules) it has. In an expert system, there are two elements—

31**a knowledge base and an inference engine. The knowledge base** includes facts and **rules that are** known to **the** system. The **inference**

engine utilizes the facts and rules to make decisions and derive new rules, which are then stored in the sys- tem to update the knowledge base. Expert systems have

166**many real-world applications such as medical diagnosis**

[29], scientific hypothesis formation [30], and structural safety monitoring [31]. The rules in an expect system can be expressed using Boolean logic or Fuzzy logic. In Boolean logic, every condition in the rules is either true or false. Fuzzy logic was introduced by Zadeh [32] to describe a degree of truth. For example, a cup with water is described as “full (1)” or “empty (0)” using Boolean logic; however, in Fuzzy logic, it can also be described using some fuzzy expressions such as “almost full”, “half full”, “near empty”. Fuzzy logic is commonly used in our daily life. An example rule in an expert system using Fuzzy logic is: IF the temperature is cold, THEN turn the heater on. In stock markets an old saying is: “buy low, sell high”. However, whether the stock value can be considered as low or high depends on the stock curves in a particular situation. Fuzzy systems have many commercial applications, such as in air conditioners, digital cameras and hand writing recognition. 2.1 Background Another representation of AI is the neural network, which mimics the processing ability of nervous systems of biological organisms (especially the human brain). Through a combination of weights and excitation functions (e.g., sigmoid functions), neural net- works can accomplish many tasks observed in humans, such as pattern recognition and image processing. Robotics is a field in which machines interact with their environment to accomplish certain tasks. While AI and robotics are not essentially connected, they are often used together to make robots appear intelligent. For example, a robot with AI can move autonomously and make decisions while interacting with the environment it is operating in. Through combining AI and robotics, machines can be created in such a way that they can learn and adapt to a changing environment. In [33], a robot was built to be capable of automatically finding compensatory behaviors after going through damage in a locomotion task. There are two common paradigms adapted for the control of a robot: deliberative paradigm and reactive paradigm. In deliberative paradigm, the robot operates on a top-down fashion and its action mainly depends on planning. A typical cycle is: sense → plan → act. In the sensing stage, the robot gets the information from the world based on sensors such as cameras, infrared sensors, etc. After pre-processing, this information would then be passed to the central control architecture that integrates all the sensing information and reasons about it.

179**Based on the** knowledge **the robot** has, it **can decide** which action **to**

take to fulfill a goal (e.g., to maximize its reward). This paradigm has led to many successful applications. The pioneering work is Shakey the robot, which is capable of reasoning about its own actions [34]. In this work the robot is operating under simplified conditions (e.g., uniform color and flat floor). Further work has been done since then to enable robots to tackle complex and changing environments [35]. In the reactive paradigm the robot makes decisions based on sense → act without deliberate reasoning or planing. Instead of building a central reasoning system to inte- grate all the sensory information, the robot could process it in parallel. Some famous robots using the reactive paradigm are Allen [36], Herbert [37] and Genghis [38]. 2.1.2 Introduction of Automation Science With the development of AI and robotics, intelligent and automation systems were commonly used to assist scientific research. Since the first clinical automated laboratory management system [39] was created in 1993, such

6**systems are increasingly used in** drug discovery, **agriculture, energy labs, etc.**

6**In order to** accelerate the **experimental process, researchers take advantage of**

intelligent and automation systems to help, for example, collect and analyze data, as this can be

3**time-consuming and** tedious **if carried out** man- ually. **The ideal situation is to make a machine** conduct scientific research **automatically without or with little human intervention.**

It could then conduct

3**experiments day and night in a constant manner without any tiredness. The** field of **automation science has been developed to a great extent because of the** in- creasing **demands of** the **drug industry and** the **relevant fields of biology and chemistry.**

High-throughput screening (HTS) systems [40] are one of the early efforts. HTS systems can, for instance, prepare, observe and analyze data automatically, thus greatly enhanc- ing the speed of experimental process. Recently, King, et al. [41, 7] have

3**built a**"Robot Scientist **"—Adam.** It **can automatically generate functional genomics hypotheses about the yeast Saccharomyces cerevisiae and carry out experiments to test and refine the hypotheses based on**

techniques developed in AI and robotics. Adam could automati- cally

3**conduct the cycles of scientific experiments: forming** a **hypothesis, initializing the experiments, explaining the results and verifying the hypothesis, and then repeating the cycle. The functional genomics hypotheses are autonomously** generated **by intelligent software and the experiments are conducted by**

coordinating different components in the automated system [41]. This "Robot Scientist" is able to conduct numerous exper- iments and observations in a day. In [42],

3**Gauld et al. developed a digital automated identification** system **(DAISY) to identify biological species automatically with high** ac- curacy **using** an **advanced image processing technique. This technology has gone though great improvement in recent years,** enhancing **the possibility of automation, or at least semi-automation, in the** process **of routine taxonomic identification. In** [43], **MacLeod et al. reported that an imaging system that** was **originally designed for identifying marine zooplankton was used by the US** government **for** automatically **monitoring** the Deepwa- ter **Horizon oil spill.**

3**They argue that taxonomists and researchers in machine learning, pattern recognition as well as** AI **should collaborate with each other in order to better**

2.2 Evolutionary Computation

3identify and name biological species. Drawing on approaches from various research areas, especially AI and

robotics, intel- ligent

6and automation systems are playing a vital role in scientific research, allowing researchers to conduct experiments more efficiently.

It is argued that the revolution of automation science may emerge in a few decades [7]. 2

172.2 Evolutionary Computation Evolutionary computation is a field studying techniques

that are inspired by biological evolution. We use

27an evolutionary computation technique to automate the generation of models during the

system identification process.

59This section is organized as follows. Section 2.2 .1

introduces biological evolution. Sec- tion 2.2.2 details the principles, strengths and weaknesses of evolutionary computation. Section 2.2.3 presents three main applications of evolutionary computation and related work. 2.2.1 Biological Evolution Biological evolution is about how living organisms evolve to adapt to environmental changes. According to Darwin's Theory of Evolution [44], individuals compete for sur- vival, and the individuals that are better adapted to the environment tend to reproduce more. This phenomenon is regarded as natural selection. From another point of view, the genes that help the species survive better would have a higher chance of being pre- served and passed on to the next generation, while the genes that are harmful would be abandoned. Natural selection tends to accumulate beneficial genetic organs [45]. Suppose that some members in a species have evolved a functional organism that is very useful (e.g., a wing that can fly). This makes it easier for these members to find food or avoid predators. The offsprings that inherited such an advantageous function are more likely to live longer and reproduce more; the other members without the advantageous function are less likely Table 2.1: The relationship between different species that coevolve. Symbols '+', '-' and '0' represent beneficial, harmful and neutral in a relationship, respectively.

3For example, "+, -" indicates A benefits and B get harmed. the

3influence of species A +,+ (reciprocity) +,0 (symbiosis) +,- (predation) the influence of species B 0,+ (symbiosis) 0,0 (neutral) 0,- (amensalism) -,+ (predation) -,0 (amensalism) -,-(antibiosis)

to reproduce. Natural selection helps the species to adapt better to their environment. At the same time, it also accelerates the extinction of the species that can not adapt well to environmental changes. For instance, the dinosaur used to be a dominant species in the past. Their size was a big advantage when the climate was mild. However, as the climate changed dramatically (e.g., extremely cold or hot), the big body was no longer an advantage as it needed too much energy. This may accelerate the extinction of dinosaurs [46]. Coevolution is a special form of

3evolution, which involves the simultaneous evolution of two or more dependent species.

A typical example of coevolution

3is fox and rabbit or parasite and host. In nature, survival

abilities of species are

3coupled. That means, the survival ability of one particular species depends not only on its

DNA (genes) grouped as chromosomes, but also on other species. For a specific species, there can be three relationships with other species: beneficial, harmful and neutral [47].

3Therefore, through permutation and combination, the relationship between two species can be summarized in Table 2.1. It includes six concrete relationship: reciprocity, neutral, symbiosis, amen- salism, predation and antibiosis

[47]. 2.2.2 Introduction of Evolutionary Computation 2.2.2.1 Principles Based on the principle of biological evolution, the genetic algorithm (GA) was proposed by Holland in the 1960s [48]. There is a population of solutions in a GA. The solution for a given problem is represented as a chromosome, which contains several genes. Each gene could be a bit, integer, or floating-point number. The GA is driven

167by a fitness function, which defines the quality of the

candidate solutions in the population. The evolutionary process is to optimize (e.g., maximize) the fitness of the individuals. There are three

10genetic operators: selection, crossover and mutation. In each generation, the

individuals with high fitness

129have a higher chance of being selected to the next generation

and producing offsprings (e.g., through crossover). Mutation can be, for example, realized by randomly changing a particular gene. Mutation in GA serves the same function as it is in biological evolution. It creates diversity in the population. There are other types of evolutionary algorithms. Fogel, Owens and Walsh invented evolutionary programming (EP) [49]. Rechenberg and Schwefel introduced evolution strategies (ES) [50, 51], which are mainly dealing with real-value continuous optimization problems. In the early 1990s, another type of evolutionary algorithm called genetic programming (GP) was presented by Koza [52]. Fig. 2.1 shows a brief classification of evolutionary algorithms. Evolutionary Algorithms

178Genetic Algorithm Evolutionary Programming Evolution Strategies Genetic Programming

Figure 2.1: Classification of evolutionary algorithms The implementation of evolutionary algorithms follows a general flow during the oper- ation process. They can be divided into five steps: initialization, evaluation, mutation, selection and termination. At the beginning, a random

142population of individuals is ini- tialized. Each individual is represented as a

string (chromosome), which contains several genes (e.g., floating point numbers). These strings encode the candidate solutions. Dif- ferent chromosomes are

> 131**then evaluated using** a predefined **fitness function. The** fitness **of** individuals **in the population**

only depends on their own chromosomes. Selection happens after evaluation of each individual, and the ones

> 105**with higher fitness are more likely to be selected to the next generation**

and have offsprings. The offsprings would go through mutation, which helps to maintain the diversity of the whole population. Fig. 2.2 show a diagram of how the evolutionary algorithms proceed. Initialization evaluation variation meet termination No criteria? selection Yes termination Figure 2.2: This diagram shows the flow of evolutionary algorithms. Adapted from [53]. As it is mentioned before, the species in nature are not independent, and they coevolve. The different kinds of relationship in Table 2.1 show that the evolution of individuals among species are coupled. Coevolutionary algorithms, which coevolve simultaneously two or more populations, are widely adapted to solve real-world problems [54, 55, 56, 57]. In principle, coevolutionary algorithms can be considered of comprising several sub- algorithms, each of which could be an evolutionary algorithm. These sub-algorithms

> 1**interact with each other** in **the fitness calculation** process. **In**

> 65**other words, the fitness of individuals in one population**

> 3**not only depends on its own chromosome, but also on the performance of other individuals** from another population **during the** coevolutionary **process.**

The essential difference between evolutionary algorithms and coevolutionary algorithms is the way of fitness evaluation. In coevolutionary algorithms, the individual's fitness de- pends on its performance and that of the individuals from the other populations. Coevo- lutionary

> 3**algorithms are generally divided into two categories—competitive** coevolution- ary algorithms **(Comp-CEA) and**

cooperative coevolutionary algorithms

> 3**(Coop-CEA).** A **Comp-CEA assesses each individual by its competitive performance with respect to its opponents, while Coop-CEA assesses each** individual **by its cooperative performance with respect to its co-operators. As discussed by Dawkins and Krebs** [58], **competitive** coevolution **can produce** the **phenomena of**"arm races" where **the** complexity **of each population** increases. **The evolution of one** population **may drive another** population **to evolve new strategies, which makes** the individuals **of** both populations **evolve a higher level of complex** behavior. **Generally, Coop-CEA is applied to the situation in which the problem can be divided into several sub-problems.**

Several cooperative populations are evolving simultaneously, and each

> 109**individual of a** sub **-population represents a part of the solution. The** fitness

of **an**

individual could be the quality of the solution formed by the combination of the individual with those from other sub-populations. In the rest of the thesis, we only discuss Comp-CEAs, which will also be referred to as coevolutionary algorithms in general. In coevolutionary algorithms, the

3**fitness of individuals is called subjective fitness** [59]. **An individual's** subjective **fitness is based on the performance of its temporary opponents** from **the**

current generation or a combination of current and past generations. The

3**fitness of individuals with the same chromosome may vary** in different generations **because of** changing **opponents.** Conversely, **the fitness in**

evolutionary algorithms

6**is called absolute or objective fitness.** Suppose **there are two populations in**

a coevolutionary algorithm.

3**One is called**"learner", **and the other is called**"evaluator". Let **L** represent **a set of learners and E** represent **a set of evaluators.**

For a learner, a simple way of calculating its subjective fitness is to count the number of evaluators (in the current population) that this learner defeated [60]. It is described in equation (2.1) as follows (reproduced from [47]): ∀i ∈ L ⇒ CFi = 1. (2.1) j∈E,∑idefeats j CFi is the

6**fitness of learner i. Another fitness calculation approach is called competitive fitness sharing** [54] **as** de- scribed **in the following** (reproduced from [47]): **∀j ∈ E**

⇒ Nj = 1. (2.2) k∈L, ∑kdefeats j ∀i ∈ L ⇒ CFi = 1 . (2.3) j∈E,∑idefeatsj Nj

3**Nj represents the number of learners that could defeat evaluator j.**

In

3**competitive fitness sharing** [54], **the learner that could defeat** the more competitive **evaluator**

gets a higher reward. For example, if learner i, in a population is the only individual to defeat evaluator j, this learner's accumulative fitness is increased by 1, as Nj is equal to 1 in Equation (2.3). The aim of using competitive fitness sharing is to preserve/award the learner that possesses

6**genetic material** that **is worth passing to the next generation. There are many ways to choose the evaluators** (temporary opponents). **Random** Pairing [61] **means finding a random temporary opponent for each** learner. **In** a **single elimination tournament** [62], **all the individuals** are **randomly** matched, **and the losers are taken out and** the **winners are selected into next** the **round of**

random matches. Alternatively, **Round Robin** [61] pairs **all evaluators**

temporarily with each learner.

3**There are also other ways such as K-random opponent** [62] **and** shared **fitness**

[54].

6**The evaluation time for random** pairing **is the shortest, but the performance is the worst; the calculation time for round robin is the longest, but the performance is the best** [47]. In **the**

coevolutionary method (Turing Learning) proposed in this dissertation, a fitness calculation similar to Round Robin is chosen. 2.2.2.2 Strengths The advantages of evolutionary computation can be summarized in the following [63]: • Conceptually Simple: Evolutionary computation techniques can be implemented using simple genetic operators (e.g., selection, crossover, mutation) as mentioned in Section 2.2.2.1. • Task-independent: Evolutionary computation techniques can be used for optimiza- tion. Many tasks in reality can be treated as function optimization or black-box optimization problems (which will be detailed in Section 2.2.3.1). • Parallel Computing: Evolutionary algorithms can run in parallel to accelerate the evolutionary process. The individual can be evaluated independently according to the fitness function. However, the selection and recombination process is normally done in series. • Robust to Changes: Evolutionary computation is robust to changes in circum- stances. Once the circumstances have changed, the evolved solutions can be used as a start for further development without the need to restart the whole process [64]. Apart from the general advantages of evolutionary computation, coevolutionary algo- rithms have the following two advantages: • Open-Ended Evolution: Coevolutionary algorithms can create an open-ended evo- lution for each population due to the complex interaction between the competing populations during the coevolutionary process. Such open-ended evolution

3**could encourage the appearance of new building blocks, thus** maintaining **the diversity of**

populations. In Darwin's natural selection, this phenomenon is referred to as "arms race" [58], which leads each species to continuously improve. • No Absolute Fitness Needed: Coevolutionary algorithms can be applied to solve problems in which absolute fitness can not be effectively defined. For example, when evolving a chess program, it is challenging to define a fitness to qualify it, although playing against fixed programs would be an option [65]. An alternative way of evaluating a chess program is making it play with competing programs and then calculating its subjective fitness [60, 66]. 2.2.2.3 Weaknesses The main weakness of evolutionary computation is that it usually requires significant (computational) efforts to obtain good solutions. In simulation, this would not necessar- ily be a problem; however, if the evaluation needs to be conducted on a physical system, it would take a long time and may also be expensive. There is no guarantee that using evolutionary computation techniques can always find the optimal or right solution. This may limit its use on some tasks that have a high demand of safety, as an occasional error may cause a system failure. Apart from the general disadvantages of evolutionary computation techniques, in par- ticular, there are three main pathologies of coevolutionary algorithms: • Red Queen Effect: When the Red Queen effect happens, two populations compete with each other and their subjective fitness keeps increasing during the coevolu- tionary process, however their objective fitness does not constantly improve. On the other hand, their objective fitness increases, but the subjective fitness does not reflect such situation [55]. • Cycling: In coevolutionary algorithms, the criteria (temporary opponents) used for evaluating an individual (solution) is changing over generations. As a consequence of this, some solutions in the previous generations may be lost and rediscovered later. This phenomenon is referred to as cycling [59]. A method of

3**overcoming the problem of**"cycling" **is**"hall of fame"[54]. "Hall of fame" retains **the** good **individuals** from **the previous generations, and**

these obtained individuals may be selected to be temporary components to evaluate the individuals from the

com- peting population in the current generation. •

3**Disengagement: During the** coevolutionary **process, when one** population **is** en- tirely **better than the other, disengagement will occur. In this case, the selection criteria** will not **make sense,**

3**since the subjective fitness of** individuals in **each** pop- ulation **is constant. The** lack **of**

selective pressure would cause each population to diverge. Common methods for addressing the disengagement problem is “resource sharing” [67] and “reducing virulence” [59]. 2.2.3 Applications of Evolutionary Computation Evolutionary computation techniques are widely used for solving various engineering tasks such as image processing, pattern recognition, robot control and system identi- fication. Many real-world applications of evolutionary computation can be found. In the area of nanophotonic light trapping, a need is the development of low cost thin film input objective function output optimizer Figure 2.3: This diagram shows the process of black box optimization. The task is to find a candidate solution (input) that can optimize (e.g., maximize or minimize) the objective function. solar photovoltaic technologies. A traditional way is fixing the structure according to an expert’s experience and trying to find good parameters. In [68], a highly efficient light-trapping structure was designed using a genetic algorithm. It was shown that this structure can increase the trapping efficiency three times compared with the classic de- sign. Another successful example is using evolutionary algorithms to design antennas for NASA’s Space Technology spacecraft [64], and one of the antennas was used in the mission. The antenna is a critical device for a spacecraft to communicate with the ground, as faulty communication may cause data loss or even a failure of the spacecraft. The antennas designed using evolutionary algorithms are significantly better than those designed by human experts. In the following sections, we will focus on three main fields in which evolutionary com- putation plays a role. 2.2.3.1 Black Box Optimization A prominent application of evolutionary computation is black box optimization. Black box optimization refers to optimizing an objective function without assuming the hidden structure (e.g., linear or differentiable) [69]. In particular, the aim is to find a set of inputs that can maximize or minimize the output of the objective function. Fig. 2.3 shows a diagram of the black box optimization. In the following, we list a few cases that evolutionary computation could be applied to solve hard optimization problems [70]. • High-dimensional:

169**As the dimension,** n, **of the** objective **function increases, the**

search space increases exponentially. This is called “curve of dimension” by Bell- man [71]. If we have to optimize a function that has 30 dimensions, and each dimension only has 20 parameter values to be selected, for a grid search in which all the possible solutions are evaluated, it will take 2030 evaluations. Suppose that each evaluation takes 1µs, the grid search would take more than 3 · 1031 years. Using evolutionary computation techniques can shorten the time to find a good solution. • Multi-Model: Multi-model means a system (function) has more than one optima (e.g., Rastrigin Function). The

5**one(s) with the best fitness value is (are)** considered **as global optima,** and **the others are** considered as **local optima.** These local **optima can**

be misleading for gradient-based search algorithms (e.g., hill climbing algorithm), as the solutions may get trapped. Evolutionary algorithms are shown to be effective to find global optima [72]. • Non-separable and non-differential: A function, g(x1, x2, · · · , xn) is non-separable, if it can not be expressed as:

136**g(x1, x2, · · · , xn) = g(x1)g(x2) · · · g(xn).**

For a separa- ble function, it would be much easier to optimize each variable separately. However, for a non-separable system, the variables can be coupled. Also, the function may not be differentiable, which makes many mathematical optimization algorithms (e.g., quasi-Newton BFGS algorithm [73] or conjugate gradient algorithm [74]) infeasible. • Multi-objective: When encountering real-world problems, we usually need to deal with multiple objectives, which need to be optimized simultaneously. For example, when

designing a car, the engineers need to consider the shape, performance of the engine and cost. As some of the objectives are conflicting, the designer should find a Pareto optimal solution [75] in which none of the values of the objective functions can be increased without decreasing the value of other objective functions. Evolutionary multi-objective optimization is a technique to generate Pareto optimal solutions. For a review, see [75]. 2.2.3.2 Evolutionary Robotics Another application of evolutionary computation is evolutionary robotics, in which the controller and/or morphology of a robot are automatically generated [76, 77]. The user considers the robot as a whole and only needs to specify the optimization criteria which are represented by a fitness function. The fitness function could be single-objective or multi-objective. The aim is to optimize the fitness using evolutionary algorithms. To optimize controllers of a robot, the normal procedure is as follows. First,

187**an initial population of random** controllers **is generated. Each**

controller is represented by a chro- mosome. If the controller is a neural network, the chromosome could be a vector of real numbers. Each controller is tested on the robot (in simulation or reality), and the robot's performance for some specific task is measured and evaluated using a pre-defined fitness function. The 'fitter' controller (i.e., with higher performance) has a higher chance of being selected and generating offspring in the next generation. This process iterates until a good solution is found. A common control structure that is adapted in evolutionary robotics is the artificial neural network due to its simple representation [78, 79]. According to the types of control strategies, different neural networks can be chosen (forward neural networks and recurrent neural networks, etc.). After selecting the structure of the robot controller, evolutionary algorithms are used for optimizing the weights of the neural networks. Note that one can even evolve the structure (topology) and weights of the neural networks [80]. There are two main research aims in evolutionary robotics [76]: 1) engineering—developing the control strategy for robots; 2) biology—to understand biological systems using sim- ulated (or physical) evolution. In engineering contexts, a range of work has been pre- sented, ranging from simple behaviors, such as phototaxis behavior [81] to complex behaviors such as navigation [82] and locomotion [83] of robots with multiple degrees of freedom. In biology contexts, evolutionary robotics is used as a tool to understand the general principles of evolution. It provides an efficient way to validate or even cre- ate hypothesis based on evolution in simulation or real robots, compared with the slow evolutionary process in nature. AVIDA [84] and AEvol [85] are two computer software systems that are used for studying the evolution of bacterias. In hypothesis validation, evolutionary robotics was used as a tool to investigate some key issues in biology. For example, whether altruism plays an important role in cooperation among species [86, 87], the conditions of emergence of communication during the evolution [88], how morphology and control are coupled [89], and the coevolution of predator and prey [90, 91]. Two methodologies are adapted in evolutionary robotics. The first one is evolving the robot's solutions in simulation and then transferring the best solution to a physical robot. The other is evolving the solutions directly on the physical robot. A drawback of the first method is the reality gap [92]. That is, the simulation used for generating the solutions may not match the robot's real operating environment, causing a reduction in performance when testing the solution in reality. Many studies aimed at reducing the reality gap [93, 94, 95]. In [93, 94], the authors presented the transferability approach to improve the control quality of a robot. The

194**controllers were** generated **in simulation, and the best**

controller (selected according to multiple criteria) was

74**transferred to the real robot,** and **the** data collected **was** used **for**

refining the simulator. This increased the qual- ity of controllers generated in simulation, and also reduced

79**the number of experiments** to be conducted **on the** real **robot.**

Some researchers also investigated implementation of evolution directly on the physical robot [82, 96, 97]. In [82], the evolution was performed entirely on the real (Khepera) robot to evolve the controller to perform a navigation task. An advantage of this is that the reality gap has been avoided; however, it may take a significant amount of time to evolve a desirable solution. In [82], it took two weeks to evaluate only 100 generations. The power supply is a problem when the robot uses a battery, as it can only last a limited time. Although the authors in [82] used a wire to connect the robot with a charging station, this is not desirable for

outdoor experiments and when using multiple robots. Recently, a distributed online onboard evolutionary method (artificial embodied evolution) has received much attention [98, 99, 100, 101]. In artificial embodied evolution, the population is distributed among different robots, and the gene exchange is done through mating. Each robot only exchanges its genes with its nearby neighbors. There is no central control over the group of robots. This ap- proach is suitable for the situation where the environment that the robots are operating in is not predictable or changing after deploying the robots. The robots need to adapt to the changing environment, while satisfying certain basic requirements inserted by the designers. A more challenging research area could be evolving both the morphology and controller of the robots in a distributed manner—evolution of things [102, 53]. This forms an open-end evolution among different artifacts, which is similar to the process of how living creatures evolve in the real world. The fast development of 3D printing makes this method appealing [103]. 2.2.3.3 System Identification System identification concerns the synthesis of models of a

> 3hidden system through con- ducting a set of experiments. It is widely used in both academia and

industry [104, 11]. The

> 3experiments are conducted by feeding a series of inputs into the system and collect- ing the outputs corresponding to these inputs. The

objective

> 3is to find a model that fits the inputs and

outputs.

> 3There are two system identification approaches: the offline approach and the online approach [104]. In the offline approach, the observed data is collected first, and the aim of modeling is to generate a model that fits the observed data. This method is often used when the data is simple to collect or the parameters and operating environment of the system do not change much in a short time. However, in some cases, parameters of the target system

may be time varying.

> 3That means the obtained model based on the previous observations can not be applied to subsequent situations. Through the

online system identification method, the model can be updated using new observed experimental data. The two approaches of system identification are shown in Fig. 2.4. Note that for some systems, there are only outputs.

> 3System identification can be divided into two main procedures: modeling and estimation. Modeling defines the order or general structure of the hidden system [106]. Estimation identifies the parameters associated with the given structure. There are many algorithms to determine the parameters for a given structure. Typical estimation algorithms are recursive prediction error methods [107], which are based on gradient search. Gradient search methods, such

as greedy algorithms, may

> 3**easily get trapped in local** optima, es- pecially **when there are numerous local** optima **near the global** optimum. An **alternative** search **method**

is using evolutionary algorithms. There are some system identification methods that combine the modeling and estimation process, for example, the NAR- MAX method [11]. Neural networks can be also used, as their structure and weights (parameters) can be optimized simultaneously. The disadvantage of neural networks for modeling is they are difficult to interpret especially when consisting of multiple layers. Target System Model (a) offline Start New input ( ) Model update Target system No Is this model good enough? Yes New output ( ) End (b) online Figure 2.4: Diagrams showing two approaches for system identification. (Ii, Oi), where i ∈ {1, 2, · · · , n}, represent pairs of input and output data. Redrawn from [105]. Genetic programming provides an alternative way for finding the structure and param- eters of the system. The models represented by genetic programming can be described in a tree-based structure. As the structure in genetic programming is evolving, the ob- tained model may be of a different structure in different runs [108]. Bloating [109] is a problem in genetic programming, where the growth of the tree increases the complexity of the model structure. Coevolutionary algorithms provide an effective way for system identification [55], [110, 111, 112, 15, 113, 114]. A range of work has been performed on simulated agents. In [110],

> 1**Bongard and Lipson proposed the estimation-exploration algorithm, a nonlinear system identification method to coevolve** inputs **and models in a way that minimizes the** number **of** inputs **to be** tested **on the system.**

In each generation, the input that leads to the highest disagreement between the models' predicted output in simulation was carried out on the real system. The

> 16**quality of the models was evaluated** by quantitatively comparing **the** output **of the** real system **and**

the models' prediction. The method was applied to evolve morphological parameters of a simulated quadrupedal

> 7**robot after it undergoes 'physical' damage. In**

a later work [111], they reported that "in many cases the simulated robot would exhibit wildly different behaviors even when it very closely approximated the damaged 'physical' robot. This result is not surprising due to the fact that the robot is a highly coupled, non-linear system; thus similar initial conditions [...] are expected to rapidly diverge in behavior over time". They

> 1**addressed this problem by using a more refined comparison metric** reported **in** [111]. **In** [112], **an**

algorithm

> 64**that is** also **based on coevolution of models and**

inputs was presented to model the simulated quadrotor helicopter and improve the control quality. The inputs were selected based on multiobjective performances (e.g., disagreement ability of models as in [110] and control quality of a given task). Models were then refined through

> 64**comparing their prediction to each selected test trajectory.** In [115], **the**

3damage detection process is conducted by a coevolutionary algorithm to extract the maximum information from the system. In [116], a coevolutionary algorithm was applied to estimate chaotic time series, in which the test data that can extract hidden information from the chaotic system co-evolves with the models.

As in other system identification methods, predefined metrics play a critical role for evaluating the performance of models. Many studies also investigated the implementation of evolution directly in physical en- vironments, on either a single robot [117, 118, 33] or multiple robots [119]. In [117], a four-legged robot was built to study how it can infer

2its own morphology through a process of continuous self-modeling. The

robot ran a coevolutionary algorithm on- board. One population evolved models for the robot's morphology, while the other evolved actions (inputs) to be conducted on the robot for gauging the quality of these models through comparing sensor data collected. In [119], a distributed coevolutionary approach was presented to coevolve on-board simulators and controllers of a group of ten robots to perform foraging behavior. Each robot has its own simulator which mod- els the environment. The evolution of each robot's simulator was driven by comparing the real-world foraging efficiency (a pre-defined fitness metric) of its nearby neighbors each executing the best controller generated by their own simulators. Each robot has a population of controllers, which evolved according to the robot's on-board simulator. The best controller was chosen for performing real-world foraging. In all of the above approaches, the model optimization is based on pre- defined metrics (explicit or implicit), which are task dependent. 2.3 Combining AI/Robotics and Animal Behavior The

3variety of animal behaviors in nature is immense, ranging from simple perception of light to complicated behaviors such as navigation and communication.

The

1scientific study of animal behavior is pursued not only because it is a subject of interest in itself, but also because the knowledge gained from it has

several practical applications. In AI/Robotics, there is a large interest in studying animal behavior, as the model/knowl- edge learned can be used to build more intelligent machines. At the same time, building a machine that mimics an animal helps researchers to better understand its behavior. In this section, we review how AI/robotics and animal behavior study benefit each other.

59This section is organized as follows. Section 2 .3.1

introduces a variety of animal behaviors observed in nature. Section 2.3.2 details how animal behavior can be used as inspiration to solve engineering tasks, especially in the area of AI/robotics. Section 2.3.3 shows how AI/Robotics can contribute to ethology, which is the (ultimate) aim of this thesis. 2.3.1 Animal Behavior in Nature Compared with the behaviors exhibited by complex animals such as mammals, insect behavior has been also widely studied by researchers. The simple neural system of insects makes it more feasible to replicate the intelligence. A basic insect behavior

3is taxis, which is its intrinsic behavioral response to a specific stimulus. Taxis is divided into different types according to the stimulus.

These behaviors

3**include phototaxis (light), chemotaxis (chemicals), thermotaxis (temperature), etc.**

An interesting taxis behavior of crickets is that the female crickets perform complex auditory orientation behavior towards the male crickets. Researchers have found this complex sound localization behavior emerges from simple reactive steering responses to specific sound pulses generated by male crickets [120]. Apart from taxis behaviors, some insects use the stimuli as cues for navigation or migration. For instance, the

3**bee uses its vision system to navigate in the air and avoid obstacles.**

Another interesting behavior found in insects is the ball movement of dung beetles [121]. Once the dung beetles form the pieces of dung into a ball, they always roll the dung-ball in a straight line using various 2.3 Combining AI/Robotics and Animal Behavior stimuli (e.g., the moon, sun and polarised light [122, 123]) as visual cues to transport the food source. This behavior ensures that they keep away from the competitors as far as possible. The stimulus-response behaviors in insects mentioned above have been investigated by biologists for centuries. Although the behavior exhibited by insects may be simple and easy to mimic, it is not trivial to be modeled and well understood [124]. When investigating such behaviors, biologists need to learn how to interact with the animal in a meaningful way to extract all of its behavioral repertoire. Apart from the behavior of a single animal,

1**swarm behaviors,** which **are emergent** (collec- tive) **behaviors that arise from the interactions of a number of animals** (especially social insects) **in a group,**

have also been widely observed in nature. The individual behaviors in a swarm tend to be relatively simple [17]. The global behavior that is exhibited in a swarm is a result of a self-organized process. Researchers found that individuals do not need the representation or complex knowledge to build a map of what the global behavior should be [125]. From the point of control, swarm behavior is a distributed control system which does not rely on central coordination. Many swarm behaviors are observed in nature. For example, the birds' flocking behavior is of particular interest to humans. This is not only because of their aesthetic shape formed in a group, but also because of the way the birds coordinate each other to maintain that shape. A simple mathematical model was proposed by [126] to describe the individual behavior of each bird in a flock. The three rules are: attraction, repulsion and alignment. In the attraction rule, the birds will be attracted by their neighbors, and this would result in each bird moving towards to the 'center' of their neighbors. Repulsion means the bird needs to avoid colliding with each other. Alignment assumes that each bird moves in the same direction with its neighbors. Although there is no proof that flocking birds follow exactly these three rules, it is attractive that boids in simulation following such simple rules can mimic the real flocking behavior closely. There are many other swarm behaviors which are also studied extensively such as the aggregation of cockroaches [127], foraging in ants [128], flashing synchronization in fireflies [129], and mound building in termites [130]. 2.3.2 Swarm Optimization and Swarm Robotics In this section, we review some techniques that are inspired by observation of animal behaviors. Section 2.3.2.1 reviews two bio-inspired algorithms— the

10**ant colony opti- mization** algorithm [131] **and** the **particle swarm optimization**

algorithm [132]. Section 2.3.2.2 introduces how the intelligence observed in animals can be applied in the field of robotics, in particular, swarm robotics. 2.3.2.1 Swarm Optimization

102**Ant Colony Optimization Ant Colony Optimization (ACO)** was **inspired by the foraging behavior of**

ants [133].

188**When an ant finds** an item of **food, it will leave**

pheromone on its path back to the nest. Other ants will be attracted by the pheromone, the strength of which may represent the quantity/quality of the food. Researchers have found that this indirect communication, which is know as stigmergy [134], leads them

> 36**to find the shortest path between** their **nest and** the location of **a food source. The** initial application **of**

ACO is to find the optimal path in a combinational (discrete) problem. Note that nowadays the application of ACO algorithms ranges from discrete optimization (e.g., routing and load balancing [135]) to continuous optimization [136]. The principle of ACO algorithms can be summarized by the two steps as follows: • Use the pheromone model (and heuristic model) to generate candidate solutions. The

> 10**pheromone model is a parameterized probability distribution** in **the search space. The**

heuristic model could be the length of the segment chosen. • The candidate solutions are used as a bias for sampling high-quality solutions in the next iteration. For a complete implementation of ACO algorithms, see [133].

> 153**Particle Swarm Optimization** 32 **Particle Swarm Optimization** (PSO) **is another** optimization **algorithm**

> 10**that is inspired by the** flocking **of birds and** schooling of **fish.**

It was first proposed by Kennedy and Eberhart [132]. The initial application is to optimize the weights of neural networks—a continuous optimization problem. It is also widely used in discrete optimization. The basic component in PSO is called a particle. A PSO algorithm consists of a finite set of particles. The movement of each particle is updated using velocity. The velocity of each particle in each time step is updated based on its current velocity, the deviation between the best position (the particle has found so far) and its current position, and the deviation between the best position by its neighbors and its current position. This usually results in the particles moving towards the high-quality solutions after certain it- erations. The update of each particle can be written using two equations as follows [132]: →−vi+1 = →−vi + c1−R→1 ⊗ (→−pi − →−xi) + c2−R→2 ⊗ (→−pg − →−xi). (2.4) →−xi+1 = →−xi + →−vi. (2.5) −→ −→ R1 and R2 are independent random number generators that return a vector of random values in range [0, 1]. c1 and c2 are referred to as acceleration coefficients. (→−pi − →−xi) and (→−pg − →−xi) represent respectively the deviation between

> 10**the best position (it has found so far) and**

its current position, and deviation between the best position by its neighbors and its current position. The first item in Eq. (2.4) keeps the particle moving in the previous direction; the second item makes the particle move towards the best position of its own; the third position forces the particle move towards to the best position that its neighbors have found. Eq. (2.5) updates the particle's position. 2.3.2.2 Swarm Robotics In the previous section, we described how swarm behaviors have inspired the design of novel optimization algorithms. In this section, we introduce how to apply swarm intelligence techniques to multi-robot research, which is referred to as swarm robotics. The behavior of many social insects (e.g, ants, termites, wraps and bees) has been used as inspirations in swarm robotics. Swarm robotics investigates how multiple robots each with limited ability communicate, coordinate, and self-organize to accomplish certain tasks. The advantages of swarm robotics are as follows: • Robustness: The robustness of a swarm robotic system can be explained in the fol- lowing: 1) A failure of a single robot does not influence other robots in the group; 2) the control is distributed; 3) as the individual is simple, it has fewer compo- nents that could fail. Note that swarm robotic systems can also be affected by errors. That is, some individuals' failure would influence the whole self-organized process [137]. • Scalability: In swarm robotics, a small change of the number of robots does not have significant impact on the global performance/behavior of the system (unless the number is sufficiently small). When investigating the performance of a swarm robotic system, a scalability study is

usually considered [138, 139]. • Flexibility: The swarm could adapt to changing tasks and generate relevant solu- tions, by changing the role of each robot [140]. In order to coordinate, robots need to interact with each other and their environment. There are three kinds of interaction in swarm robotic systems [141]. • Interaction via environment: Robots communicate by changing the environment. There is no explicit communication between each robot. In nature, the pheromone ants leave when foraging is an environmental stimulus for locating a food source. • Interaction via perception: Robots can perceive each other in a limited range. This perception is local and

174**there is no explicit communication between the robots.**

The robots can distinguish different items (e.g., other robots, objects) in the environ- ment. In nature, when ants need to collectively pull food to the nest, they need to perceive each other (to avoid collision) and the food (object). • Interaction via explicit communication: This type of communication could be re- alized by broadcast (e.g., WiFi [142]) or a distributed sensing network [143]. How to build a reliable network when the number of robots is significantly large is still a topic widely discussed. When the number of robots increases, the load of commu- nication increases exponentially. A possible solution is combining the advantage of network communication and local communication using the robots' perception.

186**A range of** tasks **have been demonstrated** in **swarm** robotics. **The**

tasks range from aggregation [144, 21, 145, 127], dispersion [146, 147], pattern formation [148, 149], col- lective movement [150] to cooperative transport [151, 152, 153, 138]. Aggregation can be considered as the fundamental behavior of other more complex tasks. In [127], a group of robots mimic the cockroaches' aggregation behavior, in which the robots join or leave a nest (place) in the environment with a probability proportional to the size of the nest. In [21], the robots each with a binary sensor were reported to aggregate into a single cluster, validated using 40 e-puck robots. The robots do not perform algorithmic compu- tation. The aggregation performance scales well with 1000 robots in simulation. In [154], Werfel et al. designed a group of termite-inspired robots that work collectively to build several structures. The robots communicate with each other using stigmergy. In [150], a group of nine Kobot robots were reported to mimic the flocking behavior of birds. These robots followed some simple rules similar to those proposed by Reynolds [126]. In cooperative transport, Chen et al. [138] presented a strategy

154**for a group of** miniature **robots to transport** a tall **object** to **the**

goal. In the task, the robots only push the object when the robots' vision of the

11**goal is occluded by the object. This strategy**

was

11**proved to be able to** push any **convex object to the goal** in **a**

2D environment. 2.3.3 Contribution of AI/Robotics to Ethology In the previous sections, it was reviewed how the study of animal behavior can be used as inspiration for designing optimization algorithms and robotic systems. However, AI/robotics can also benefit the study of animal behavior. This section reviews two approaches: learning from synthesis and robot-animal interaction. 2.3.3.1 Learning from Synthesis Ethologists have studied animal behavior for over a century.

3**There are some basic steps that ethologists follow in the study of animal** behavior [155]. **The first step is observation, and after that they formulate some scientific questions on the observed** behavior, **and generate** hypotheses **to answer these questions.**

They then

3**conduct related experiments** on **the animals and collect data. After** analyzing **the data, the conclusion will be made to support or reject their hypothesis.**

Robotics or artificial life can be used as an alternative methodology to investigate and understand animal behavior.

143**Robots can be used as** physical **models of animal** behaviors for testing **hypotheses**

[156, 6]. For example, taxis behavior has often been implemented on mobile robotic systems for investigating steering and navigation [124]. In [157], Webb used a robot to model the phonotaxis behavior of crickets [158]. The robot can locate the position of a sound source and moves towards it under different conditions. There was good agreement between data collected from experiments with the robot and animals.

3**Another taxis** behavior **—chemotaxis, in which animals follow a specific chemical** trail, **has been used as a model for robots to find** odor sources **based on artificial neural networks** [159] **and even Braitenberg vehicles**

[160]. Robots can be used as a validation for the models obtained from biologists and allow them to better understand the animal behavior from a synthetic point of view. Besides, roboticists can generate new hypotheses and test them using (simulated or physical) robots. In a social behavior study, Balch et al. [161] built executable models of the behaviors of ants and monkeys, which can be directly executed by multi-robot systems.

3**The aim is to** show **how** research into **multi-robot** systems **can contribute to the** study of collective **animal**

behaviors. In [162],

3**Chappell et al. argue that there are many ways in which biologists interested in natural intelligence can learn from AI and** robotics, and **they outline specific kinds of contributions that AI can make to biological** studies. **They also give some suggestions on how AI** and robotics **researchers** could **collaborate with biologists to**

solve some cutting- edge problems in animal behavior. As opposed to the works mentioned above, the method proposed in this thesis aims to synthesize models of agent (animal) behaviors automatically, rather than manually. This could help to spare scientists from having to perform numerous laborious experiments, allowing them instead to focus on using the generated models to produce new hypotheses. 2.3.3.2 Robot–Animal Interaction

3**Besides pure robot-based or AI research, researchers also use robots to interact with real animals. They build and** program **robots** (i.e., replicas) **that can be inserted into the group of social animals**

[163, 164, 165, 166, 167].

1**Robots can be created and** sys- tematically **controlled in such a way that they are accepted as con- or** hetero-specifics **by the animals in the group**

[168].

> 6**In this case, one**"animal" **in a group is completely controlled and they can observe the** behaviors **of the mixed society** [169]. **The** behavior **of the inserted robot can be controlled and the model can also be embedded into the robot for verification** [170]. **The** behavior **of robots can be programmed in such a way that its** behavior **is not influenced by other animals in the group, and they can be used as demonstrators or leaders in the experiments. Further more, it is** simpler **to** test **a hypothesis through controlled interaction in social** behaviors. **In**

[163], a replica fish which resembled the appearance (i.e., visual morphology) of stick- lebacks was created to investigate two types of interaction: recruitment and leadership. In [171], a robot-fish that can interact intelligently with live zebrafish to study their preference and locomotion behavior was designed. In [165], autonomous robots which executed a derived model were mixed into a group of cockroaches to modulate their decision-making of selecting shelter in the aggregation behavior. The robots behaved in a similar way to the cockroaches. Although the robots' appearance was different to that of the cockroaches, the robots released a specific odor that the cockroaches could detect and regard the robots as conspecifics. In [172, 167],

> 3**Vaughan et al.** have **built a mobile robot that can interact with** ducks **in a circular arena and drive them to the safe place.**

In [173],

> 3**Gribovskiy et al. designed a robot capable of interacting with chicks to study how the** behavior **of chicks** is **influenced by the others in a group. Although robots which are well designed can be mixed** with **social animals, building such** kinds **of** robots **is a time-consuming process. It is also expensive to some extent and requires the collaboration of researchers from different disciplines.**

In the aforementioned works, the models were manually derived and the robots were only used for model validation. This robot-animal interaction framework could be enhanced through the system identification method proposed in this thesis, which autonomously infers the collective behavior. 3 Reverse Engineering Swarm Behaviors Through Turing Learning As mentioned in Chapter 1,

> 1**swarm behaviors are emergent behaviors that arise from the interactions of a number of** simple individuals **in a group**

[17]. Researchers in artificial intelligence and swarm robotics use simulated agents or physical robots to mimic and understand the swarm behaviors

> 7**observed in nature, such as** foraging, **aggregation,**

and flocking. This approach is what we refer to as understanding from synthesis. It has been used to validate models of swarm behaviors, provide insights, or even generate new hypotheses [165]. The design of controllers for the simulated agents or physical robots usually adheres to similar conditions to those found in biological swarm systems. For example, the controllers are distributed, that is, there is no central control for the swarm; the structure of the controllers is relatively simple. In this chapter, we demonstrate how the proposed system identification (coevolutionary) method

1allows a machine to infer the behavioral rules of a group of homogeneous agents in an autonomous manner. A replica, which resembles the agents under investigation in terms of behavioral capabilities, is

mixed into the group. The coevolutionary algorithm consists of

1two competitive populations: one of models, to be executed on the replica, and the other of classifiers. The classifiers observe the

motion of an individual1 in the swarm for a fixed time interval. They are not, however, provided with the individ- ual's sensory information. Based on the individual's motion data, a classifier outputs a Boolean value indicating whether the individual is believed to be an agent or replica. The classifier gets a reward if and only if it makes the correct judgment. The

1fitness of the classifiers thus depends solely on their ability to discriminate between the

agents 1Note that 'individual/robot' in the context of swarm behaviors refers to a single unit. It could refer to an agent under investigation or a replica executing a model. However, in the context of evolutionary algorithms (see Section 3.1.1.3), 'individual' refers to a chromosome. 3 Reverse Engineering Swarm Behaviors Through Turing Learning and the replica.

1Conversely, the fitness of the models depends solely on their ability to 'trick' the classifiers into categorizing them as an agent. Consequently, our

method does not rely on predefined metrics for measuring the similarity of behavior between models and agents; rather, the metrics (classifiers) are produced automatically in the learning process. Our method is inspired by the Turing test [174, 175], which machines can pass if behaving indistinguishably from humans. Similarly, the models, which evolve, can pass the tests by the coevolving classifiers if behaving indistinguishably from the agents. We hence call our method Turing Learning. To validate the Turing Learning method,

7we present two case studies on canonical problems in

swarm robotics:

7self-organized aggregation [21] and object clustering

[22]. We show that observing individual motion is sufficient

7to guide the learning of these collective behaviors.

In this chapter, we only present the simulation results. A real- world validation using physical robots based on one of the case studies will be presented in Chapter 4.

16This chapter is organized as follows. Section 3 .1 describes the implementation of Turing Learning and the

two swarm behaviors under investigation. Section 3.2 introduces the simulation platform and setups for performing coevolution runs. Section 3.3 presents the results obtained from the two case studies. In particular, Section 3.3.1 analyzes the evolution of models, objectively measuring their quality in terms of

local and global behaviors. Section 3.3.2 investigates the coevolutionary dynamics. Section 3.3.3 investigates the evolution of classifiers, showing how to construct a robust classifier system to potentially detect abnormal behaviors in the swarm. Section 3.3.4 studies the effect of observing only a subset of agents in the swarm. Section 3.3.5 presents a study where an aspect of the agents' morphology (their field of view) and brain (controller) are inferred simultaneously. Section 3.3.6 shows the results of using Turing Learning to learn other swarm behaviors. Section 3.3.7 presents a study showing the method's sensitivity to noise. Section 3.4 summaries the findings in this chapter. 3.1 Methodology In this section, we present the Turing Learning method and two case studies: self- organized aggregation [21] and object clustering [22]. In both case studies, individuals 3.1 Methodology execute simple behavioral rules that lead to meaningful emergent behaviors on a global level. 3.1.1 Turing Learning Turing Learning uses a coevolutionary algorithm that comprises two populations:

> 1**one of models, and one of classifiers.** These populations **coevolve competitively.** In **the** following, we describe **the** models, **classifiers,**

optimization algorithm, fitness calculation method and termination criterion. 3.1.1.1 Models We assume to have a replica that has actuation and sensing abilities that are equivalent to those of the agents under investigation. In other words, the replica must be able to produce behavior that—to an external observer (classifier)—is indistinguishable to that of the agent. For example, if the observer tracks the position of a bird in three dimensions (3D), the replica (which could be artificial or real) must be able to produce 3D motion data. There can be one or more replicas. In Chapters 3 and 4, the replica(s) will be mixed into a group of homogeneous agents. They should therefore be perceived by the agents as con-specifics [165]. In Chapter 5, a single replica is studied in isolation. Models are executed on replicas. The models can be represented explicitly (e.g., param- eters) or implicitly (e.g., artificial neural networks). 3.1.1.2 Classifiers The classifier can be any algorithm that takes a sequence of data about the observed individual as input, and outputs a decision (i.e., whether the observed individual is believed to be an agent or replica). We represent the classifier as a recurrent Elman neural network [176], which is shown in Figure 3.1. The network has i inputs, h

> 7**hidden neurons and one output neuron.**

Figure 3.1: The plot shows the structure of a classifier, which

> 1**is a recurrent Elman**

neural network, with i inputs, h hidden neurons, and one output neuron (O). O is used for making a judgment. Two bias neurons (which are not shown) with a constant input of 1.0 are connected to each neuron of the hidden and output layers. See text for details.

> 1**Each neuron of the hidden and output layers has a bias. The network has a total of**

(i + 1)h + h2 + (h + 1) parameters,

> 1**which all assume values in R. The activation function used in the hidden and the output neurons is the logistic sigmoid function, which has the range (0, 1) and is defined as: sig x = 1 1** + e **−x**∀x ∈ **R.**

(3.1) The classifier makes judgments about all individuals—both agents and replica. It ob- serves each individual over a set period of time (i.e., a

> 1**trial). The final value of** its **output neuron, O, is used to make** the **judgment: the classifier**

judges the individual to be a replica if O < 0.5, and an agent otherwise. The classifier's memory (value of hidden neurons) is reset after each judgment. 3.1.1.3 Optimization Algorithm The optimization of models and classifiers is realized using evolutionary algorithms. In this thesis, we use a

> 2**(μ+λ) evolution strategy with self-adaptive mutation strengths**

[177, 109] to optimize either population. The optimization algorithm

> 1**can be thought of** con- sisting **of two sub-algorithms: one for the** model population, **and another for the** classifier population. **The**

sub-algorithms

> 2**do not interact with each other except for the fitness calculation step (described in** Section **3.** 1.1.4). **In** each sub **-algorithm,**

the value of μ and λ is equal, which is half of the population size. The implementation of the evolutionary algorithm is detailed as follows.

> 2**In this algorithm, an individual is a 2-tuple, a = (x, σ), where x**∈ Rn **represents** ob- jective **parameters, and σ ∈ (0, ∞)n represents mutation strengths. The i-th mutation strength in σ corresponds to the i-th element in x.**

> 2**Each generation g comprises a population of μ** individuals: **P(g) = a(** 1g), **a(** 2g), . . . , **a(**μg) . { } **In the population of the first generation, P(0),** all the objective **parameters are** initial- ized **to 0.0 and all the** mutation **strengths are initialized to 1.0. Thereafter, in** every **generation g, the μ parent individuals are first used to create λ offspring individuals by recombination. For the generation of each recombined individual a′** k **(g), k**

∈

> 2**{1, 2, . . . , λ}, two individuals are chosen randomly, with** replacement, **from the parent population:** aχ(g) and **a(**ψg), where **χ,**

ψ ∈ {1, 2, . . . , μ}. The intermediate population, P′(g), is produced us- ing discrete and intermediate recombination, which generates the

> 2**objective parameters and the mutation strengths of the recombined individual,** respectively: **x′k(,** gi) = **x(**χg,)i OR **x(**ψg **,)i,**

(3.2) σk′(,gi) = σ(χg,i) + σψ(g

> 2**,)i /2, (3.** 3) ( ) **where i ∈ {1, 2, . . . , n} is indexing the elements within the vectors and, in** Equation 3.2, **the selection is performed randomly and with equal probability. Each of the λ recombined individuals is then mutated in order to obtain the final offspring population, P′′(g).** This is **done according to: σ′k′,(**

ig) = σk′(,gi) exp (τ ′Nk (0, 1) + τ Nk

9,i (0, 1)) , x′k′(, ig) = x′k(,

gi) + σk″,(ig)Nk,i (0, 1) , (3.4) (3.5)

2for all {k, i}, where k ∈ {1, 2, . . . , λ} is indexing the individuals within the population and i ∈ {1, 2, . . . , n} is indexing the elements within the vectors. Equation 3.4 generates the perturbed mutation strength from the original one according to a log-normal

distri- bution. Equation 3

2.5 mutates the objective parameter according to a normal distribution having the perturbed mutation strength as its deviation. In

Equation 3.4, Nk

2(0, 1) and Nk ,i (0, 1) are both random numbers generated from a standard normal distribution; however, the former is generated once for each individual (i.e. for each value of k), while the latter is generated separately for each element within each individual (i.e. for each combination of k and i). The parameters τ ′ and τ determine the learning rates of the mutation strengths, and are set as τ ′ = 1/2 2n, τ = 1/2 2 n (similar to

[178]), where √ √ n corresponds to the population size. √

2Once the offspring population has been generated, the μ individuals with the highest fitness

2from the combined population, P(g)∪P″(g) (which contains μ+λ individuals), are selected as the parents to form the population of the next generation, P(g+1). Individuals with an equal fitness have an equal chance of being selected. 3. 1.1.4 Fitness Calculation Let the

population sizes for the models and classifiers in the coevolution be M and C, respectively. Let the number of replicas and agents in a trial be nr and na, respectively. nt trials are conducted for a model in each generation; throughout this thesis, we assume nt = 1. In our thesis, whenever multiple replicas are used, each of them executes a different model. The fitness of each model in a trial is determined by

2each of the C classifiers in the competing population.

1For every classifier that wrongly judges the model as an agent, the model's fitness increases by 1. After all evaluations, the model's fitness takes a value in {0, 1, 2, . . . , C}. This value is then normalized to [0, 1]. The

7fitness of each classifier in a trial is

determined by its judgments for the nr replicas (each executing a different model) and na agents. For each

1**correct judgment of the** replica, **the classifier's fitness increases by** 2n1r ; **for each correct judgment of the** agent, **the classifier's fitness** increases **by** 2n1a . **Therefore, the fitness of each classifier in** a **trial is in** range **[0,**

1]. Mnr trials2 are conducted in each generation, and the fitness value of each classifier is t ┌hen┐ normalized to [0, 1]. 3.1.1.5 Termination Criterion The coevolutionary algorithm stops after running for a fixed number of generations. 3.1.2 Case Studies 3.1.2.1 Problem Formulation The agents used in the case studies of Chapters 3 and 4 are embodied and move in a two- dimensional, continuous space. The agents' embodiment is based on the

1**e-puck** [179], **which is a miniature,** differential **-wheeled robot.**

Figure 3.2 shows an e-puck robot used in the physical experiments in Chapter 4. Each agent is equipped with

1**a line-of-sight sensor that** it can use **to** detect **the**

item (e.g., the background, other agents or objects [21], [22]) in front of it. The swarm behaviors investigated in this thesis use a reactive control architecture, as found in many biological systems3. The motion of each agent solely depends on the state of its line-of-sight sensor (I). Each possible sensor state, $I \in \{0,1,\cdots,n-1\}$, is mapped

1**onto a pair of** predefined **velocities for the left and right wheels,**

$(v\ell I, vrI)$. $v\ell I, vrI \in$

8**[−1,1] represent the normalized left and right wheel velocities, respectively, where 1 (−1) corresponds to the wheel rotating** forwards **(backwards) with maximum**

velocity. Given n sensor states, any reactive behavior can thus be represented using 2We suggest choosing nr to be a factor of M. 3For example, researchers have found that the complex auditory orientation behavior of female crickets is derived from simple reactive motor responses to specific sound pulses [120]. Figure 3.2:

15**An e-puck** robot **fitted with a black 'skirt' and a top marker**

for motion tracking. 2n system parameters. In the remainder of this thesis, we describe the corresponding controllers by writing the 2n parameters as a tuple in the following order: $p = (v\ell 0, vr0, v\ell 1, vr1, \cdots, v\ell(n-1), vr(n-1))$. (3.6) We assume that the replica has the same differential drive and line-of-sight sensor4 as the agents. The system identification task is thus to infer the control parameters in Equation (3.6). This explicit representation makes it possible for us to objectively measure the quality of the obtained models in the post-evaluation analysis (as discussed in the results section). To make the evolution more challenging, the search space for the algorithm in simulation is unbounded. That is, the model parameters are unconstrained, and the replica can move with arbitrary speed. The classifier does not have any prior knowledge about the individual under investiga- tion. It is fed with the sequence of motion data of the individual. It has two input neurons (i = 2),

1**five hidden neurons** (h = 5) **and one output neuron. The**

input neu- rons represent the linear speed (v) and angular speed (ω) of the individual. They are obtained by tracking the positions and orientations of individuals. In simulation, the 4In Section 3.3.5, we show that this assumption can be relaxed by also evolving some aspect of the agent's morphology. initial configuration after 60 s after 180 s after 300 s Figure 3.3: Snapshots of the aggregation behavior of 50 agents in simulation. tracking is noise-free (situations with noise being present are considered in Section 3.3.7 and the physical experiments in Chapter 4 where noise is inherently present). We define the linear speed to be positive

1when the angle between the individual's orientation and its direction of motion is smaller than

π/2 rad, and negative otherwise. In the following, we detail the behavioral rules of the two swarm behaviors. 3.1.2.2 Aggregation

7In this behavior, the sensor is binary,

that is, n = 2.

1It gives a reading of I = 1 if there is an agent in the line of sight, and I = 0 otherwise. The environment is

1free of obstacles. The objective for the agents is to aggregate into a single compact cluster as fast as possible.

Further details, including a validation with

740 physical e-puck robots, are reported in

[21]. initial configuration after 20 s

1after 40 s after 60 s Figure 3.4: Snapshots of the object clustering behavior in simulation. There are 5 agents (blue)

and 10 objects (green). The 'optimal'

1controller for aggregation was found by performing a grid search over the entire space of possible controllers (with finite resolution) [21]. The

'optimal' controller's parameters are: p = (−0.7, −1.0, 1.0, −1.0) . (3.7)

1When I = 0, an agent moves backwards along a clockwise circular trajectory

(vℓ0 = −0.7 and vr0 = −1.0).

1When I = 1, an agent rotates clockwise on the spot with maximum angular velocity

(vℓ1 = 1.0 and vr1 = −1.0). Note that rather counter-intuitively, an agent never moves forward, regardless of I. With this controller, an agent provably aggregates with another agent or a quasi-static cluster of agents [21]. Figure 3.3 shows snapshots from a simulation trial with 50 agents. 3.2 Simulation Platform and Setups

3.1.2.3 Object Clustering This behavior uses n = 3 sensor states:

> 5I = 0 if the sensor is pointing at the background (e.g., the wall of the environment,

if the latter

> 1is bounded), I = 1 if the sensor is pointing at an object, and I = 2 if it is pointing at another agent. The objective of the agents is to arrange the objects into a single compact cluster as fast as possible. Details of this behavior, including a validation using 5 physical e-puck robots and 20 cylindrical objects, are presented in [22]. The

controller's parameters, found using an evolutionary algorithm [22], are: p = (0.5, 1.0, 1.0, 0.5, 0.1, 0.5) . (3.8)

> 1When I = 0 and I = 2, the agent moves forward along an anti-clockwise circular trajectory, but with different linear and angular speeds. When I = 1, it moves forward along a clockwise circular trajectory.

Figure 3.4 shows snapshots from a simulation trial with 5 agents and 10 objects. 3.2 Simulation Platform and Setups

> 146In this section, we present the agent platform for simulating the

two swarm behaviors under investigation and the simulation setups for performing coevolution runs. 3

> 1.2.1 Simulation Platform We use the open-source Enki library

[180], which models the kinematics and dynamics of rigid objects, and handles collisions. Enki

> 8has a built-in 2-D model of the e-puck. The

robot is represented

> 5as a disk of diameter 7. 0 cm and mass 150 g. The inter-wheel distance is 5.1 cm. The speed of

each wheel can be set independently. Enki induces noise on each wheel speed by multiplying the set value by a number in the range (0.95, 1.05) chosen randomly with uniform distribution. The

> 113maximum speed of the e-puck is 12.8 cm/s, forward or backward. The

line-of-sight sensor is simulated by casting a ray from the e-puck's front and checking the first item with which it intersects (if any). The range of this sensor is unlimited in simulation. In the object clustering case study, we model objects

> 5as disks of diameter 10 cm with mass 35 g and a coefficient of static friction with the ground of 0.58,

which makes it movable by a single e-puck. The robot's control cycle is updated every 0.1 s, and the physics is updated every 0.01 s. 3.2.2 Simulation Setups In all simulations, we used an unbounded environment. For the aggregation case study, we used groups of 11 individuals—10 agents and 1 replica that executes a model. The initial positions of individuals were generated randomly in a square region of sides 331.66 cm, following a uniform distribution (average area per individual = 10000 cm2). For the object clustering case study, we used groups of 5 individuals—4 agents and 1 replica that executes a model—and 10 cylindrical objects. The initial positions of individuals and objects were generated randomly in a square region of sides 100 cm, following a uniform distribution (average area per object = 1000 cm2). In both case studies, individual starting orientations were chosen randomly in [−π, π] with uniform distribution. We performed 30 coevolution runs for each case study. Each run lasted 1000 generations. The model and classifier populations each consisted of 100 solutions (μ = 50, λ = 50). In each trial, classifiers observed individuals for

7**10 s at 0.1 s intervals**

(100 data points). 3.3 Simulation Results 3

7.**3.1 Analysis of Evolved Models**

In order to objectively measure the quality of the models obtained through Turing Learn- ing, we define two metrics. Given an evolved controller (model) x and the original agent 3.3 Simulation Results

1**1.5 value of model parameters 1.0** 0.5 **0.**

0 -0.5 -1.0 -1.5 vℓ0 vr0 vℓ1 model parameter (a) Aggregation vr1

7**value of model parameters** -0.5 -1 **.0** 1 **.5 1.**

0 0.5 0.0 -1.5

7**vℓ0 vr0 vℓ1 vr1 vℓ2 vr2**

model parameter (b) Object Clustering Figure 3.5:

1**Parameters of the evolved models with the highest subjective fitness in the**

1000th generation in the coevolutions for (a) the aggregation behavior and (b) the object clustering behavior. Each box corresponds to 30 coevolution runs in simulation. The dotted black lines correspond to the values of the parameters that the system is expected to learn (i.e., those of the agent). controller p, where x, p ∈ [−1, 1]2n, we define the absolute error (AE) in a particular parameter i ∈ {1, 2, . . . , 2n} as: AEi = |xi − pi|. (3.9) We define the mean absolute error (MAE) over all parameters as: MAE = 1 2n 2n ∑i=1 AEi . (3.10)

1**Fig. 3.** 5 **shows a box** plot5 **with the parameters of the evolved models with the highest subjective** fitness6 **in the final generation.** It **can be seen** that Turing Learning **identified the**

parameters for both behaviors with good accuracy (dotted black lines represent the ground truth, that is, the parameters of the observed swarming agents). In the case of aggregation, the means (standard deviations) of the AEs in the parameters were (from left to right in Fig. 3.5(a)): 0.01 (0.01), 0.01 (0.01), 0.07 (0.07) and 0.06 (0.04). In the case of object clustering, these values were: 0.03 (0.03), 0.04 (0.03), 0.02 (0.02), 0.03 (0.03), 0.08 (0.13) and 0.08 (0.09). We also investigated the evolutionary dynamics. Fig. 3.6 shows how the model pa- rameters converged

1**over generations. In the aggregation** case study, **the** parameters

corresponding to l = 0 were learned **first.**

After around 50 generations, both vℓ0 and vr0

1**closely approximated their true values (−0.7 and −1.0),**

shown in Fig. 3.6(a).

1**For l = 1, it took about** 200 **generations for** both vℓ1 **and** vr1 **to converge.**

A likely reason for this effect is that an agent spends a larger proportion of its time seeing nothing (l = 0) than other agents (l = 1)—simulations revealed these percentages to be 91.2% and 8.8%, respectively (mean values across 100 trials). In the object clustering case study, the parameters corresponding to

1**l = 0 and l = 1** were **learned faster than the parameters**

corresponding to l = 2,

1**as shown in Fig.** 3.6 **(b). After about 200 generations,**

vℓ0, vr0, vℓ1 and vr1

1**started to converge; however it took about** 400 **generations for** vℓ2 **and** vr2 **to approximate their true values.**

Note that an agent spends the highest proportion of its time seeing nothing (l = 0), followed by objects (l = 1) and other agents (l = 2)—simulations revealed these proportions to be 53.2%, 34.2% and 12.6%, respectively (mean values across 100 trials). Although the evolved models approximate the agents well in terms of parameters, it 5The

1**box plots presented here are all as follows. The line inside the box represents the median of the data. The edges of the box represent the lower and the upper quartiles**

2**of the data,** whereas **the whiskers represent the lowest and the highest data points that are within 1.5 times the range from**

7**value of model parameters 1.** 5 **1** 0.5 **0**

-0.5 vℓ0 vr0 vℓ1 vr1 -1

7**-1.5 value of model parameters**

-0.5 -1 -1.5 1.5 1 0.5 0 1 200 (a) Aggregation 400 600 generation 800 1000

7**vℓ0 vr0 vℓ1 vr1 vℓ2 vr2**

1 200 400

1600 800 1000 generation (b) Object Clustering Figure

3.6: Evolutionary process of the evolved model parameters

1for (a) the aggregation behavior and (b) the object clustering behavior. Curves represent median values across 30 coevolution runs.

Dotted black lines indicate true values. has often been observed in swarm systems that small changes in individual agent be- haviors can lead to vastly different emergent behaviors, especially with large numbers of agents [182]. For this reason, we evaluated the quality of the emergent behaviors that the models give rise to. In the case of aggregation, a good measure of the emergent behavior the

1lower and the upper quartiles, respectively. Circles represent outliers.

6The

1fitness of the models depends solely on the judgments of the classifiers from the

competing population, and is hence referred to as subjective. 800 700 dispersion 600 500 400 0 5 10 15 20 25 30 index of controller (a) Aggregation 800 700 dispersion 600 500 400 0 5 10 15 20 25 30 index of controller (b) Object Clustering Figure 3.7: (a) Dispersion (after 400 s) of 50 agents executing the original aggregation controller (red box) or one of the 30 evolved models (blue boxes) of the 1000th generation. (b) Dispersion (after 400 s) of 50 objects in a swarm of 25 agents executing the original object clustering controller (red box) or one of the 30 evolved models (blue boxes). In both (a) and (b), boxes show distributions over 30 trials. The dotted black lines indicate the minimum dispersion that 50 agents/objects can possible achieve [181]. See Section 3.3.1 for details. is the dispersion of the swarm after some elapsed time as defined in [21]7. For each of the 30

1models with the highest subjective fitness in the final generation,

we performed 30 7The measure of dispersion is based on the robots'/objects' distances from their centroid. For a formal definition, see Equation (5) of [21], Equation (2) of [22] and [181]. trials with 50 agents each executing the model. For comparison, we also performed 30 trials using the original controller (see Equation (3.7)). The set of initial configurations was the same for all models and the original controller. Fig 3.7(a) shows the dispersion for the original controller and models after 400 s. All models led to aggregation. We performed a statistical test8 on the final dispersion of the agents between the original controller and each model.

15There was no statistically significant difference in 26 out of

30 cases (30 out of 30 cases with Bonferroni correction). In the case of object clustering, we use the dispersion of the objects after some elapsed time as a measure of the emergent behavior. With the original controller (see Equa- tion (3.8)) and each of the models, we performed 30 trials with 25 agents and 50 ob- jects. The results are shown in Fig. 3.7(b). In a statistical test on the final dispersion of the objects between the original controller and each model,

15there was no statistically significant difference in 24 out of

30 cases (26 out of 30 cases with Bonferroni correction). We also investigated the evolutionary process of the model parameters. Figure 3.6

1shows the convergence of the model parameters over generations. In the

aggregation behavior, the parameters **corresponding to l = 0** were learned **first.**

After about 50 generations, both vℓ0 and vr0

1**closely approximated their true values (−0.7 and −1.0)**

shown in Figure 3.5(a).

1**For l = 1, it took about** 200 **generations for** both vl1 **and** vr1 **to converge.**

A likely reason for this effect is that an agent spends a larger percentage of its time seeing nothing (l = 0) than other agents (l = 1)—simulations revealed these percentages to be 8.8% and 91.2%, respectively (mean values over 100 trials).

1**In the object clustering behavior, the parameters** corresponding **to l = 0 and l = 1** were **learned faster than the other two parameters**

corresponding to l = 2, as shown in the Figure 3.6(b). After about 200 generations, vℓ0, vr0, vl1 and vr1

1**started to converge; however it took about** 400 **generations for** vl2 **and** vr2 **to approximate their true values. This**

is likely because an agent spends the most percentage of its time seeing nothing (l = 0), followed by objects (l = 1) and other agents (l = 2)—simulations revealed these percentages to be 53.2%, 34.2% and 12.6% , respectively (mean values over 100 trials). 8Throughout this thesis, the statistical test used is a two-sided Mann-Whitney test with a 5% signifi- cance level.

98**1 0.8 fitness value 0.6 0.4 0.2 0 1**

0.8

1**fitness value 0.** 6 **0.** 4 **0.** 2 **0 average fitness of models best fitness of models average fitness of classifier best fitness of classifier** 1 **200 400 600 800 1000 generation (a) Aggregation**

1**average fitness of models best fitness of models average fitness of classifier best fitness of classifier** 1 **200 400 600 800 1000 generation**

(b) Object Clustering Figure 3.8:

1**This plot shows the subjective fitness of the classifiers and the models for (a) the** aggregation **behavior and (b) the** object **clustering** behavior. **The curves show the** median value **across 30** coevolution **runs.**

3

1**.3.2 Coevolutionary Dynamics In order to analyze how the classifiers and the models interact with each other during the course of the coevolution, we investigate the dynamics of the subjective fitness of the classifiers and the**

models as shown in Figure 3.8. In the aggregation behavior, at the beginning, the fitness of the classifiers is 0.5 as they output 1 for all the agents and models, which means the classifiers make uninformed decisions9. Therefore, the fitness of the models starts from 1.0, as all the classifiers judge them the agent. Then, the average fitness of the classifiers quickly increases, corresponding to the decline of the average fitness of the models. As the models learn to adapt, the average fitness of the classifiers only increases slightly until about the 50th generation. After that, the average fitness of the models starts to increase. However, the best fitness of the

1classifiers is still higher than that of the models. The

best

1fitness of the models surpasses that of the classifiers after about the 120th generation; at this point, the fitness of the 'best'

model (selected by the classifiers)

1is around 0.7. This means that the'best' model is able to mislead 70% of the classifiers into judging it as the agent. From the 200th generation onwards, the fitness of the classifiers and models remains"balanced" until the last generation. The coevolutionary dynamics of the object clustering behavior is similar. Compared with the dynamics of

the aggregation behavior, it takes more generations for the

1fitness of the models to surpass that of the classifiers.

1This could be explained by the higher number of parameters and the higher complexity of the behavior to be evolved

in the object clustering behavior. 3.3.3 Analysis of Evolved Classifiers The primary outcome of the Turing Learning method (and of any system identifica- tion method) is the model, which has been discussed in the previous section. However, the evolved classifiers can also be considered as a useful byproduct. For instance they could be used to detect abnormal agents in a swarm. We will now analyze the perfor- mance of the evolved classifiers.

156For the remainder of this chapter, we consider only the aggregation case

study. To assess the performance of the classifiers, we measure the percentage of correct judg- ments over agents and a wide range of models. The models are uniformly distributed 9Note

1that in the implementation of the coevolutionary algorithm, the parameters of the models and classifiers are initialized to 0. 0, which means all the classifiers are identical at the 1st generation.

1 0.9 decision accuracy 0.8 0.7 0.6 0.5 0.4 best classifier (subjective) classifier system (subjective) best classifier (archive) classifier system (archive) best classifier (decision) 12 6 10 60 100 600 1000 generation

Figure 3.9: The average decision accuracy of the best classifiers and classifier systems over generations (nonlinear scale) in 30 coevolution runs. The error bars show standard deviations. See text for details. 1 0.9 0.8

46**fitness value 0.7 0.6 0.5 0.4 best**

classifier (subjective) best classifier (archive) best classifier (decision) best classifier (objective) 12 6 10 60 100 600 1000

140**generation Figure 3.10:** This plot shows **the average fitness of the**

classifiers over generations (non- linear scale) in 30 coevolution runs. The fitness is calculated based on 14641 testing models. See text for details. across the entire parameter space of the agents: [−1, 1]4. To keep the analysis of clas- sifiers within a reasonable computation time, we discretize this space using 11 settings per parameter, to obtain: X = {−1.0, −0.8, ..., 0.8, 1.0}4. This discretized space is a grid consisting of |X | = 114 = 14641 points (i.e., models). The classifier's performance is computed as follows. The model is executed by a replica mixed into a group of 10 agents (as in the coevolution runs). 10 trials are performed using a set of initial configurations common to all classifiers. The motion data is fed to each classifier, which makes 10 judgments per individual. If the classifier consistently judges the individual as a model (i.e. not an agent) in 10 out of 10 trials, it outputs a "model" decision. Otherwise, it outputs "agent". This conservative approach was used to minimize the risk of false positive detection of abnormal behavior. 3.3.3.1 Using a Single Classifier The average decision accuracy of the classifier with the highest subjective fitness in 30 coevolution runs is shown in Fig. 3.9 (best classifier (subjective)). The accuracy combines the percentage of correct judgments about models (50% weight) with the percentage of correct judgments about agents (50% weight). The accuracy of the classifier increases in the first 5 generations, then drops and fluctuates within range 62%–80%. An alternative strategy is to select the classifier that achieves the highest fitness when evaluated on the whole historical tracking data (not just those of the current generation). The decision accuracy of this classifier is also shown in Fig. 3.9 (best classifier (archive)). The trend is similar to that of best classifier (subjective). The accuracy increases in the first 10 generations, and then starts decaying, dropping to around 65% by the 1000th gen- eration. However, in the earlier generations, the accuracy of the best classifier (archive) is higher than that of the best classifier (subjective). For a comparison, we also plot the highest decision accuracy that a single classifier achieves for each generation (best classifier (decision)). Interestingly, the accuracy of the best classifier (decision), which is shown in Fig. 3.9 (black curve), increases almost monotonically, reaching a level above 95%. Note that to select the best classifier (decision), one needs to perform additional trials (146410 in this case). At first sight, it is counter-intuitive that selecting the best classifier according to the historical data still leads to low decision accuracy. This phenomenon, however, can be explained when considering the model population. We have shown in the previous section (see especially Fig. 3.6(a)) that the models converge rapidly at the beginning of the coevolutions. As a result, when classifiers are evaluated in later generations, the trials are likely to include models very similar to each other. Classifiers that become Agent one 1.5 1.0 ● output 0.5 0.0 ● −0.5 0 20 40 60 80 100 time step Agent three 1.5 1.0 ● ● output 0.5 0.0 ● ● −0.5 0 20 40 60 80 100 time step Agent two 1.5 1.0 ● ● output 0.5 0.0 ● ● −0.5 0 20 40 60 80 100 time step Model 1.5 1.0 output 0.5 0.0 ● −0.5 0 20 40 60 80 100 time step (a) decision-making process Agent one activation value 1.5 1.0 0.5 ● 0.0 ● ● ● ● ● −0.5 0 20 40 60 80 100 time step Agent three activation value 1.5 1.0 0.5 ● 0.0 ● ● ● ● ● −0.5 Agent two activation value 1.5 1.0 0.5 ● ● 0.0 ● ● ● ● −0.5 0 20 40 60 80 100 time step Model activation value 1.5 1.0 0.5 ● 0.0 ● ● ● ● ● −0.5 0 20 40 60 80 100 0 20 40 60 80 100 time step time step ● h1 h2 h3 h4 h5 (b) activation of hidden neurons Figure 3.11: This plot shows the (a) decision-making process and (b) the corresponding activation value of the 5 hidden neurons of a classifier for three random- chosen agents and the replica that executes a very good model in a trial. Hidden neurons are labeled as h1, h2, h3, h4, and h5. overspecialized to this small set of models (the ones dominating the later generations)

191**have a higher chance of being selected** in **the**

post-evaluation. These classifiers may however have a low performance when evaluated across the entire model space. To investigate whether the decision accuracy is a good measurement of the quality of the classifiers, we plot a figure showing the fitness of the best classifiers corresponding to Figure 3.9. The fitness is calculated based on the 14641 models. In addition, we also plot the classifier with the highest fitness for comparison, which is termed as best classifier (objective). The results are shown in Figure 3.10. As we can see, the trend of fitness curves show good correspondence to that of the decision accuracy in

Figure 3.9, although there is still a small gap between the best classifier (decision) and best clas- sifier (objective). This means measuring the quality of classifiers according to decision accuracy is a reasonable choice. In order to understand how the classifiers make judgment, we analyze the internal pro- cessing through monitoring the activation of hidden neurons. Figure 3.11 shows the decision-making process and the corresponding activation value of the 5 hidden neurons of the classifier with the highest decision accuracy in the last generation of a coevolution run for 3 randomly-chosen agents and the replica in a trial. The model executed on the replica has a parameter set of (−0.7, −1.0, 1.0, −0.9), which is very near to that of the agent in Equation (3.7). As we can see in Figure 3.11(a), for the agents, the classifier outputs 0 at the beginning, and after a certain time period outputs 1. This means the classifier needs some time to make the correct judgment. Note that for some time steps after it started to output 1, it still outputs 0, but this happens only occasionally. For the model, it always outputs 0. This phenomenon can be explained by the activation value of the hidden neurons of the classifier. The classifier's activation value of the hidden neurons (labeled as h1, h2, h3, h4, and h5 in Figure 3.11(b)) reveals how it processes the motion of different individuals10. It seems that the decision-making ("sudden jump" in Figure 3.11(a)) of the classifier is related with the variation of h3 and h4. Consider the classification of the agents for example, at the beginning, h3 has a very low value (which is almost 0) and then after some time it starts to increase. The variation tendency of h4 is opposite. Every time the classifier's judgment starts to jump from 0 to 1, the value of h3 starts to surpass that of h4. h1, h2 and h5 are almost not activated during the whole period, although sometimes the activation of h2 influences the judgment of the classifier. For instance, there are two peaks at about 50th and 80th time step in h2 for agent two and three, which correspond to a wrong judgment of the classifier (that is, 10Note that the activation of hidden neurons may not be the same for all the classifiers due to the symmetry of neural networks, but the dynamics is similar. it outputs 0) at the same point. For the model, all the hidden neurons are almost not activated. 3.3.3.2 Using a Classifier System The results of the previous section have shown that using a single classifier is not a good solution; although there may be a good classifier in each generation, it may take significant effort (trials with agents) to find it.

150**To address this problem, we propose the use of**

a classifier system, that is, a number of classifiers working in tandem to judge a given candidate. We choose the best 10 classifiers11. This is either the set of 10 classifiers

1**with the highest** subjective **fitness in the** current **generation** or **the**

set of classifiers that achieve the highest fitness when evaluated on the whole historical data. If one or more classifiers make a decision about the candidate as a model (i.e. not an agent), the system outputs a "model" decision. Otherwise, it outputs "agent". The results of using a classifier system are shown in Fig. 3.9 (green and magenta, re- spectively). The two systems exhibit significantly improved decision accuracy across all generations. After 1000 generations, each system has a high accuracy of above 95%, on average. When post-evaluating the performance of classifier systems, we kept the setup the same as the one used in the coevolution runs (i.e., 10 agents and 1 replica in each trial). In practice, the ratio of agents and replicas in a trial could be different. Therefore, we analyzed the scalability of the classifier systems selected in the last generation over 30 coevolution runs with different number of agents in the group. Note that there is always one replica in the group. We tested 14641 different models. Figure 3.12 shows the 11Note that the number of classifiers chosen to form the system here is not necessarily optimal. There is no guarantee that the individually best few classifiers will form the best system when working in tandem. In principle, one could exhaustively search every possible combination of a given number of classifiers from the population. However, this is often infeasible—with our settings of choosing 10 classifiers out of 100, 1.73 × 1013 possibilities exist. We therefore propose the heuristic of choosing the individually best classifiers to form the system, and empirically show that this nevertheless yields good results. 1.0 0.8 decision accuracy 0.6 0.4 0.2 0.0 1 5 10 50 100 number of agents Figure 3.12: This plot shows the decision accuracy of the classifier system (archive) in the 1000th generation of 30 coevolution runs for different numbers of agents in the group. A single replica was present in each trial. In each case, 14641 different models were tested. decision accuracy of the classifier system (archive) when changing the number of agents in the group. As we can see, the decision accuracy of the classifier system is not affected by the variation of the number of agents, which shows the robustness of the classifier system. As we mentioned before, we chose 10 classifiers to form a classifier system to have a reasonable decision accuracy. Here we investigate how the classifier systems perform with various number of classifiers chosen to form a system. The decision accuracy of the classifier system (archive) is shown in Figure 3.1312. It seems that as long as the number of classifiers chosen to form a system is within a certain range, the system could perform well. For example, when the number of selected classifiers is between 10 and 25, the classifier system can obtain a high decision

accuracy. However, when the number is high (e.g., 75), the decision accuracy declines dramatically. 12The result of classifier system (subjective) is similar. 1.0 0.8 decision accuracy 0.6 0.4 0.2 0.0 1 2 5 10 25 50 75 100 number of classifiers Figure 3.13: This plot shows the decision accuracy of the classifier system (archive) in the 1000th generation of 30 coevolution runs with various number of classifiers chosen to form a system. 0.5 observing all agents observing one agent 0.4 MAE 0.3 0.2 0.1 0.0 1 5 10 50 100 number of agents Figure 3.14: MAE (defined in Equation (3.10)) of the

7**evolved models with the highest**

subjective fitness after 1000 generations, when using Turing Learning with varying numbers of agents (excluding the replica). Red and blue boxes show, respectively, the cases where all agents are observed, and one agent is observed. Boxes show distributions over 30 coevolution runs. 3.3.4 Observing Only a Subset of Agents So far, we have used motion data about all agents in the swarm when evaluating classifiers. However, this may not always be feasible in practice. For instance, given a video recording of a large and/or dense swarm, extracting motion data about all agents may be infeasible or lead to highly inaccurate results. A more practical solution might be to only track a subset of agents (e.g., by equipping them with markers). We now compare the case where all agents are observed with the other extreme, where only one agent is observed. We study how these two cases compare as the swarm size increases. We conducted 30 coevolution runs with each number of agents $n \in \{1, 5, 10, 50, 100\}$. There was always one replica in the group. When observing only one agent, this was chosen randomly in each trial. Note that the total number of trials in each coevolution run for the case of observing all agents and one agent is identical. We measured the total square error of the model

1**with the highest subjective fitness in the** last (1000th) **generation**

of each run. The results are

25**shown in Fig.** 3.14. **There is no** statistically **significant difference**

for any n. On the other hand, as the swarm size increases, performance improves. For example,

190**there is a statistically significant difference between** n = 10 **and**

n = 100, both when observing all agents and one. These results suggest that the key factor in the coevolutionary process is not the number of observed agents, so much as the richness of information that comes from increasing inter- agent interactions, and is reflected in each agent's motion. This means that in practice, observing a single agent is sufficient to infer accurate models, as long as the swarm size is sufficiently large. A similar phenomenon was observed in a different scenario, where the goal was to distinguish between different 'modes' of a swarm (i.e., global behaviors) through observing only a few individuals [183]. I= 0 ?/2 no fragments of ?/2 agents within this sector (a) I= 1 ?/2 ?/2 (b) Figure 3.15: A diagram showing the angle of view of the agent's sensor investigated in Section 3.3.5. 3.3.5 Evolving Control and Morphology In the previous sections, we assumed that we fully knew the agents' morphology (i.e., structure), and only their behavior (controller) was to be identified. We now present a variation where one aspect of the morphology is also unknown. The replica, in addition to the four controller parameters, takes a parameter $\theta \in [0, 2\pi]$ rad, which determines the horizontal

31**field of view of its** sensor, **as shown in Fig.**

3.15 (however, the sensor is still binary). Note that in the previous sections the agents' line-of-sight sensor can be considered as a sensor with angle of view of 0 rad. The models now have five parameters. As before, we let the coevolution run in an unbounded search space (i.e., now, R5). However, as θ is necessarily bounded, before a model was executed on a replica, the parameter corresponding to θ was mapped to the range $[0, 2\pi]$ using a logistic sigmoid function (Equation (3.1)). The controller parameters were directly passed to the replica. In this setup, the classifiers observed the individuals for 100 s in each trial (preliminary results indicated that this setup requires a longer observation time). Fig. 3.16(a) shows the parameters of the subjectively best models in the last (1000th) generations of 30 coevolution runs. The

means (standard deviations) of the AEs in each model parameter were: 0.02 (0.01), 0.02 (0.02), 0.05 (0.07), 0.06 (0.06) and 0.01 (0.01). 3

1value of model parameters 2 1 0 -1 -2

3

1value of model parameters 2 1 0 -1 -2

vℓ0 vr0 vℓ1 vr1 θ model parameter (a) vℓ0 vr0 vℓ1 vr1 θ model parameter (b) Figure 3.16: Parameters (controller parameters and angle of view in rad) of the evolved models with the highest subjective fitness in the 1000th generation corre- sponding to the case of the agents' angle of view equal to (a) 0 rad and (b) π/3 rad. Boxes show distributions over 30 coevolution runs. Dotted black lines indicate true values. All parameters including θ were still learned with high accuracy. The case where the true value of θ is 0 rad is an edge case, because given an arbitrarily small ε > 0, the logistic sigmoid function maps an infinite domain of values onto (0, ε]. This makes it easier for the coevolution to learn this parameter. For this reason, we also considered another scenario where the agents' angle of view is π/3 rad rather than 0 rad. 6 value of model parameters 4 2 0 -2 -4 6 value of model parameters 4 2 0 -2 -4 vl0 vr0 vl1 vr1 θ 1 200 400

1600 800 1000 generation (a) vl0 vr0 vl1 vr1

θ 1 200 400 600 800 1000 generation (b) Figure 3.17: Evolutionary process of the evolved models (controller parameters and an- gle of view in rad) in the 1000th generation corresponding to the case of the agents' angle of view equal to (a) 0 rad and (b) π/3 rad. Boxes show distri- butions over 30 coevolution runs. Dotted black lines indicate true values.. The controller parameters for achieving aggregation in this case are different from those in Equation (3.7). They were found by re-running a grid search with the modified sensor. Fig. 3.16(b) shows the results from 30 coevolutions with this setup. The means (standard deviations) of the AEs in each parameter were: 0.04 (0.04), 0.03 (0.03), 0.05 (0.06), 0.05 (0.05) and 0.20 (0.19). The controller parameters were still learned with good accuracy. 400 frequency 300 200 100 0 0.00 0.02 0.04 0.06 0.08 0.10 MAE (a) 2.0 1.5 AE 1.0 0

7.5 0.0 vℓ0 vr0 vℓ1 vr1

model parameter (b) Figure 3.18: This plot shows (a) a histogram of the MAE (defined in Equation (3.10)) of the evolved models and (b) the AEs (defined in Equation (3.9)) of each model parameter in the 1000th generation over 1000 random behaviors. For each behavior, we performed one coevolution run. In (a), 43 points that have MAE larger than 0.1 are not shown. The accuracy in the angle of view is noticeably lower, but still reasonable. 3.3.6 Evolving Other Behaviors The aggregation controller that agents used in our case study was originally synthesized by searching over the space [−1, 1]4, using a metric to assess the swarm's global per- formance [21]. Other points in this space lead to different global behaviors that can be 'meaningful' to a human observer (e.g. circle formation [139]). We now investigate whether our coevolutionary method can learn arbitrary controllers in this space, irrespective of the global behaviors they lead to. We generated 1000 controllers randomly in [−1, 1]4, with uniform distribution. For each controller we per- formed one coevolution run, and selected the subjectively best model in the last (1000th) generation. Fig. 3.18(a) shows a histogram of the MAE of the evolved models. The distribution has a single mode close to zero, and decays rapidly for increasing values. Over 89% of the 1000 cases have an error below 0.05. This suggests that the accuracy of Turing Learning is not highly sensitive to the particular behavior under investigation (i.e., most behaviors are learned equally well). Fig. 3.18(b) shows the AEs of each model parameter. The means (standard deviations) of the AEs in each parameter were: 0.01 (0.05), 0.02 (0.07), 0.07 (0.6) and 0.05 (0.20). We performed a statistical test on the AEs between the model parameters corresponding to I = 0 (vℓ0 and vr0) and I = 1 (vℓ1 and vr1). The AEs of the evolved vℓ0 and vr0 were significantly lower than those of the evolved vℓ1 and vr1. This was likely due to the reason reported in Section 3.3.1; that is, an agent spends more time seeing nothing (I = 0) than other agents (I = 1) in each trial. 3.3.7

1Noise Study We conducted a study to investigate how the performance of Turing Learning is affected by noise on the measurements of the individuals'

position and orientation. As the e- puck robot has a maximum linear speed of 12.8 cm/s, the maximum distance that it can travel in one control cycle (0. 1 s) **is 1.28 cm. For this reason, we define a disturbance of 1.28 cm to an** individual's **position as a measurement error of 100%. Similarly, we**

define 3.4 Summary 1.2 1.0 0.8 MAE 0.6 0.4 0.2 0.0 0 25 50 75 100 magnitude of noise (percentage) Figure 3.19: This plot shows the MAE (defined in Equation (3.10)) of the evolved pa- rameters when increasing the percentage of noise on the measurements of the individuals' position and orientation in the aggregation behavior. Each box corresponds to 30 coevolution runs. See text for details. a

1**100% measurement error on the** individual's **orientation as the maximum change in orientation that an e-puck can undergo in one control cycle. This corresponds to 0.5 rad. We conducted 5 sets of 30 coevolutions for the noise** values **M = {0, 25, 50, 75, 100}%. In a coevolution with a noise value M , every measurement of an** individual's posi- tion **is** perturbed **in a random direction and by a random distance chosen uniformly in 0, 1.28** 1M00 **cm. Similarly, every**

orientation

1**measurement is perturbed by a random** ang[le **chosen]uniformly in −0.5**

1M00 , 0.5 1M00 . Figure 3.19 shows MAE (defined in Equation (3.10)) of the evolved parameters in the [ ]

1**final generations of the coevolutions. This plot reveals that the system still performs** rel- atively **well when a considerable amount of noise** affects **the** individuals' **motion tracking system. 3.**

4 Summary This chapter has presented the simulation results of using the Turing Learning method to autonomously learn swarm behaviors through observation. To our knowledge, Turing Learning is the first system identification method that

1**does not rely on any** predefined **metric to** quantitatively **gauge the** difference **between** agents **and**

learned models. This eliminates the need to choose a suitable metric and the bias that such metric may have on the obtained solutions. Through competitive coevolution of models and classifiers,

1**the system successfully learned two swarm behaviors (self-organized aggregation and object**

clustering). Both the model parameters, which were automatically inferred, and emergent global behaviors closely matched those of the original swarm system. We also constructed a robust classifier sys- tem that, given an individual's motion data, can tell whether the individual is an original agent or not. Such classifier system could be effective in detecting abnormal behavior, for example, when faults occur in some members of the swarm. Note that Turing Learn- ing produces these classifiers automatically without the need to define a priori what constitutes abnormal behavior. A scalability study showed that the interactions in a swarm can be characterized by the effects on a subset of agents. In other words, when learning swarm behaviors especially with large number of agents, instead of considering the motion of all the agents in the group, we could focus on a subset of agents. This becomes critical when the available data about agents in the swarm is limited. Our approach was proven to work even if using only the motion data of a single agent and replica, as the data from this agent implicitly contained enough information about the interactions in the swarm. In this chapter, the model was explicitly represented by a set of parameters. The evolved parameters could thus be compared against the ground truth, enabling us to objectively gauge the quality of evolved models in

two case studies as well as for 1000 randomly sampled behaviors. In principle, we could also evolve the structure of the agent's control system. The results of learning the agent's angle of view showed that our method may even learn the morphology of the swarming agents. In collective behavior, abnormal agent(s) may have a great impact on the swarm [137]. For the same reason, the insertion of a replica that exhibits different behavior or is not recognized as con-specific may disrupt the global behavior and hence the models obtained may be biased. An appropriate strategy would be to isolate the influence of the replica. In particular, to evaluate a model one could perform two trials, one with only replicas each executing the model and the other with only agents. The data of the replicas and 3.4 Summary agents from each trial could then be fed into the classifiers for making judgments. Some preliminary results suggest that there is no significant difference between either approach for the case studies considered in this chapter. In Chapter 5, we use Turing Learning to infer the behavior of a single agent. In all cases that are considered, either the agent or the replica is present. 4 A Real-World Validation of Turing

> 152**Learning In this chapter, we** present **a** real-world validation **of** Turing **Learning.** In particular, **we**

have built an autonomous system for performing system identification with little human intervention. We demonstrate how the system can be used to identify the behavior of a swarm of real agents. The agents and replica are represented by physical robots. We

> 79**use the same type of robot** (e-puck) **as in** simulation. **The**

agents execute the aggregation behavior described in Section 3.1.2.2. The replicas execute the evolved models. We use two replicas to speed up the coevolutionary process, as will be explained in Section 4.4. Compared with simulations, which usually do not model very accurately the dynamics (such as collision), evolutions that are embodied on physical systems break the gap between simulation and reality and may bring us new insight on how to implement evolution in real world.

> 16**This chapter is organized as follows. Section 4.1** introduces **the**

physical platform, which includes the robot arena, the robot platform and the sensors implementation. Section 4.2 details the tracking system, including motion capture and video processing. Section 4.3 describes the programs executed by each component (machine, agent and replica) during the coevolutionary learning process. Section 4.4 describes the experimental setup. Sec- tion 4.5 discusses the results obtained. This includes the analysis of the evolved models and the analysis of the evolved classifiers. Section 4.6 analyzes the sensitivity of Turing Learning for individual failure during the experimental process. Section 4.7 summaries the results obtained and discusses the findings in this chapter. 4 A Real-World Validation of Turing Learning Agents video stream (robot positions & orientations) Overhead Camera Replica model updates Computer start/stop signal Robot Arena Figure 4.1: Illustration of the general setup for inferring the behavior of physical agents— e-puck robots (not to scale). The computer runs the coevolutionary algo- rithm, which produces models and classifiers. The models are uploaded and executed on the replica. The classifiers run on the computer. They are pro- vided with the agents' and replica's motion data, extracted from the video stream of the overhead camera. 4.1 Physical Platform The physical setup, shown in Figure 4.1, consists of an arena with robots (representing agents or replicas), a personal computer (PC) and an overhead camera. The PC runs the coevolutionary algorithm1. It communicates with the replicas, providing them models to be executed, but does not exert any control over the other agents. The overhead camera supplies the PC with a video stream of the swarm. The PC performs video processing to obtain motion data about individual robots. We will now describe the physical platform in more detail. 4.1.1 Robot Arena The robot arena was rectangular with sides 200 cm × 225 cm, and bounded by walls 50 cm high. The floor had a light gray color, and the walls were painted white. 1The evolution of the model population could in principle be conducted on the on-board micro- controller of the e-puck, but running it on the PC reduces experimental time [82] and eases post- evaluation analysis. 4.1 Physical Platform Left? wheel #5 #6 #4 #7 7.0?cm #0 #3 #1 ?Camera #2 Infrared

> 11**?sensors Figure** 4 **.2:** Schematic **top-view of an e-puck, indicating the locations of its** motors, **wheels, camera and** infrared **sensors.**

Note that the marker is pointing to- wards the robot's back. 4.1.2 Robot Platform and Sensor Implementations 4.1.2.1 Robot Platform In Section 3.2.1, we presented the e-puck's shape and dimensions as a basis for the agents' embodiment in simulation. We now present further details about the e-puck relevant to our physical implementation. A schematic top view of the e-puck, showing the sensors and actuators, is shown in Figure 4.2. The e-puck has a on-board micro-controller, which is the Microchip dsPIC30F6014A with 8KB of

> 8**RAM and 144 KB of flash memory. The**

two wheels are driven by the stepper motors.

> 22**The e-puck has** a directional **(color) camera located** in **its front. The** resolution of the **camera**

is up to 640 × 480, corresponding to the horizontal and vertical angle view of 56◦ and 48◦. There are eight infrared proximity sensors around the body of the robot. The proximity sensors are only used for collision/wall avoidance in the physical coevolutions where the environment is bounded. 4.1.2.2 Sensor Implementations We implemented the line-of-sight sensor using the e-puck's directional camera. For this purpose, we wrapped the robots in black 'skirts' (see Figure 3.2) to make them distin- guishable against the light-colored arena. However, we use the camera in monochrome mode, and sub-sample the image to 40 × 15 pixels, due to the e-puck's limited memory (which cannot even store a single full-resolution image). While in principle the sensor could be implemented using one pixel, we used a column of pixels from a sub-sampled im- age to compensate for misalignment in the camera's vertical orientation. The gray values from these pixels were used to distinguish robots (I = 1) against the arena (I = 0). If any pixel of that column exceeds a certain threshold in its gray scale, the sensor outputs 1; otherwise, it outputs 0. For more details about this sensor realization, see [21]. We also used the e-puck's infrared sensors, in two cases. Firstly, before each trial, the robots dispersed themselves within the arena (behavior R2 in Section 4.3.2). In this case, they used the infrared sensors to avoid both robots and walls, making the dispersion process more efficient. Secondly, we observed that using only the line-of-sight sensor can lead to robots becoming stuck against the walls of the arena, hindering the coevolutionary process. We therefore used the infrared sensors for wall avoidance,

> 4**but in such a way as to not affect inter-robot** interactions2. **In the** following, we details **the**

implementation of these two programs (disperse program and wall avoidance program). 1) The implementation of disperse program after a trial:

> 4**After finishing a trial, we disperse the robots for a while (which is equivalent to initial configuration for a new trial) in order to automate the coevolutionary learning process. This program consists of two behaviors obstacle avoidance and disperse. The obstacle avoidance behavior is to prevent the robots colliding with other robots and the walls. In particular, before executing the disperse behavior, each robot detects whether some other objects (robots/walls) exist around it using its infrared proximity sensors. If it detects something, it moves away from the objects through adjusting the linear and angular speed accordingly using a single-layer neural network controller. The obstacle avoidance behavior lasts for 3 seconds. In the disperse behavior, each robot is moving forward with a fixed linear speed while avoiding collisions with other robots and the walls. This behavior lasts for 5 seconds.** 2) **The implementation of wall avoidance program during a trial:** During **a** trial, **in** order to reduce **the**

4**chances of robots getting stuck against the walls** 2To do so, **the**

e-pucks determine whether a perceived object is a wall or another robot. 4.2 Motion Capture and Video Processing

4**(note that the aggregation behavior was designed in an unbounded environment), we imposed a wall avoidance effect to the original behavior, but in such a way as to not affect inter-robot interactions. In particular, when the robot detected the white walls using the infrared sensors or saw another robot (I=1) using the camera, it executed the same behavior. For example, for the agent, it would also turn on the spot when detecting the walls, which makes it easier to avoid the walls. However, the behavior of the replica depends on the model it is executing. Different from the Disperse program, the program of wall avoidance was only triggered when the value of any of the robot's infrared sensor was above a pre-set high threshold. This ensures that the value of the robot's infrared sensors when other robots (covered with a black 'skirt') were nearby was always below the threshold. Therefore, the wall avoidance program did not affect the aggregation of robots.**

4.2 Motion Capture and Video Processing 4.2.1 Motion Capture To facilitate motion data extraction, we fitted robots with markers on their tops, con- sisting of a colored isosceles triangle on a circular white background. The triangle's color allowed for distinction between robots; we used blue triangles for all agents, and orange and purple triangles for the two replicas. The triangle's shape eased extraction of robots' orientations. Note that the orientation of the triangle is pointing to the backward of the e-puck robot so that it can be easily attached. The robots' motion was captured using a GigE color camera (Basler Technologies), mounted around 300 cm above the arena floor. The camera's frame rate was set to 10 fps. The video stream was fed to the PC, which performed video processing to extract motion data about individual robots (position and orientation). The size of the arena in the image is 700 pixels × 800 pixels. An image of the arena captured using the overhead camera is shown in Figure 4.3. Figure 4.3: An image of the robot arena captured by the overhead camera. The robots inside the arena are covered with a colored triangle for facilitating the motion tracking. 4.2.2 Video Processing The video processing software was written using OpenCV—an open-source computer vision library [184]. The

137**details of the** video processing **algorithm are described as follows. The** image captured using **the**

overhead camera was encoded using RGB. We changed the encoding into HSV in order to make the tracking algorithm less sensitive to lighting variation. This was realized using a built-in function inside the OpenCV library. After that, the image was converted into grayscale and thresholded. As the background of the arena was light-colored, there was no need to do background extraction. After that, the color images was converted into binary images. A morphological operation (erosion followed by dilation) is applied on the binary images to filter some noise. Blobs in the binary image with a size above certain threshold (36 pixels) are used for robot detection. These selected blobs indicate the robots in the arena. Figure 4.4 shows the selected blobs in an image. 4.3 Coevolution with Physical Robots Agents Replica one Replica two Figure 4.4: The binary images showing the blobs of the robots (agents or replicas) in the arena after video processing. The blobs are selected according to certain compactness and size. The robots were tracked using the nearest neighbor algorithm. For each blob in the image, we found a minimum triangle that encloses the marker, and obtained the coor- dinates of the three vertices of this triangle. The vertex that has the longest distance to the other two vertices indicates the

175**direction of the robot.** Therefore **the** orientation **of the robot**

was estimated using the

185**vector pointing from the midpoint of the** other two vertices **to**

this vertex. We use moments [185] to calculate the position of the center of the robot. The x and y coordinate of the position is the 1th

85**order spatial moments around x-axis and y-axis** divided by **the 0th order central moments of the**

blob, respectively. Figure 4.5 shows a diagram of the video processing algorithm. 4.3 Coevolution with Physical Robots To automate the coevolutionary process, the robots (agents and replicas) were implicitly synchronized. This was realized by making each robot execute a fixed behavioral loop of constant duration. The PC also executed a fixed behavioral loop, but the timing was determined by the signals received from the replicas. Therefore, the PC was synchronized with the swarm. The PC communicated with the replicas via Bluetooth. At the start of a coevolution run, or after a human intervention (see Section 4.4), robots were initially synchronized using an infrared signal from a remote control. acquisition from camera image input (RGB) color conversion color image (HSV) threshold binary image blob selection blobs tracking positions & orientations output Figure 4.5: A diagram showing the flow of image processing algorithm used in the tracking system.

181**Figure 4.** 6 **shows a** flow **diagram of the** programs run by **the**

PC and the replicas, respec- tively. Dotted arrows indicate communication between the units. The agents executed a similar behavioral loop to the replicas. In the following, we detail the

4**states of the programs executed by the** PC, **replicas, and agents.**

4.3.1 PC Program • P1: Wait for “Stop” Signal. The program is paused until “Stop” signals are received from both replicas. These signals indicate that a trial has finished. • P2: Send Model Parameters. The PC sends new model parameters to the buffer of each replica. • P3: Wait for “Start” Signal. The program is paused until “Start” signals are received from both replicas, indicating that a trial is starting. 4.3 Coevolution with Physical Robots PC Replica P1: Wait for "Stop" Signal R1: Send "Stop" Signal (1s) P2: Send Model Parameters R2: Disperse (8s) R3: Receive Model Parameters (1s) P3: Wait for "Start" Signal R4: Send "Start" Signal (1s) P4: Track Robots R5: Execute Model (7s) P5: Update Coevolutionary Algorithm Figure 4.6: Schematic of the programs run by the PC and a replica in the physical experiments. Dotted arrows represent communication between the two units. See Section 4.3 for details. • P4: Track Robots. The PC waits 1 s and then tracks the robots using the overhead camera for 5 s. The tracking data contains the positions and orientations of the agents and replicas. • P5: Update Coevolutionary Algorithm. The PC uses the motion data from the trial observed in P4 to update the fitness of the corresponding two models and all classifiers. Once all models in the current generation have been evaluated, the PC also generates new model and classifier populations. The fitness calculation and optimization algorithm are described in Section 3.1.1.4. The PC then goes back to P1. 4.3.2 Replica Program • R1 : Send “Stop” Signal. After a trial stops, the replica informs the PC by sending a “Stop” signal. The replica waits 1 s before proceeding with R2, so that all robots remain synchronized. Waiting 1 s in other states serves the same purpose. • R2 : Disperse. The replica disperses in the environment,

4**while avoiding collisions with other robots and the walls. This behavior lasts**

8 s. • R3 : Receive Model Parameters. The replica reads new model parameters from its buffer (sent earlier by the PC). It waits 1 s before proceeding with R4. • R4 : Send “Start” Signal. The replica sends a start signal to the PC to inform it that a trial is about to start. The replica waits 1 s before proceeding with R5. • R5 : Execute Model. The replica moves within the swarm according to its model. This behavior lasts 7 s (the tracking data corresponds to the middle 5 s, see P4). The replica then goes back to R1. 4.3.3 Agent Program • The agents follow the same behavioral loops as the replicas. However, in the states analogous to R1, R3, and R4, they simply wait 1 s rather than communicate with the PC. In the state corresponding to R2, they also execute the Disperse behavior. In the state corresponding to R5, they execute the original

aggregation controller, rather than a model. 4.4 Experimental Setup As in simulation, we used a population size of 100 for classifiers (μ = 50, λ = 50). However, the model population size was reduced from 100 to 20 (μ = 10, λ = 10), to shorten the experimental time. We used 10 robots: 8 representing agents executing the original aggregation controller (Equation (3.7)), and 2 representing replicas that executed models. This meant that in each trial, 2 models from the population could be evaluated simultaneously; consequently, each generation consisted of 20/2 = 10 trials. The coevolutionary algorithm was implemented without any modification to the code used in simulation (except for model population size and observation time in each trial). 4.5 Results We still let the model parameters evolve unboundedly (i.e., in R4). However, as the speed of the physical robots is naturally bounded, we applied the hyperbolic tangent function (tanh x) on each model parameter, before sending a model to a replica. This bounded the parameters to (−1, 1)4, with −1 and 1 representing the maximum backward and forward wheel speeds, respectively. The coevolution runs proceeded autonomously. In the following cases, however, human intervention was made: • The robots had been running continuously for 25 generations. All batteries were replaced. • Hardware failure occurred on a robot, for example because of a lost battery con- nection or because the robot became stuck on the floor. Appropriate action was taken for the affected robot to restore its functionality. • A replica lost its Bluetooth connection with the PC. The connection with both replicas was restarted. • A robot indicated a low battery status through its LED after running for only a short time. That robot's battery was changed. After an intervention, the ongoing generation was restarted, to limit the impact on the coevolutionary process. 4.5 Results We conducted 10 coevolution runs using the physical system. Each run lasted 100 gener- ations, corresponding to 5 hours (excluding human intervention time).

> 22**Video recordings of the** coevolution runs **can be found in the online supplementary** materials [186]. **4.**

5.1 Analysis of Evolved Models We will first investigate the quality of the models obtained.

> 161**To select the 'best'** model from **each** coevolution **run, we post-**

evaluated all models of the final generation 5 times

> 1**value of model parameters 1.** 5 1 **.0 0.5**

0.0 -0.5 -1.0 -1.5 vℓ0 vr0 vℓ1 vr1 model parameter Figure 4.7: Parameters of the evolved models in the 100th generation of 10 physical coevolution runs. Dotted black lines indicate true values. using all classifiers of that generation. The parameters of these models are shown in Figure 4.7. The means (standard deviations) of the AEs in each parameter were: 0.08 (0.06), 0.01 (0.01), 0.05 (0.08), and 0.02 (0.04). To investigate the effects of real-world conditions on the coevolutionary process, we performed 10 simulated coevolution runs with the same setup as in the physical runs. Figure 4.8 shows the evolutionary dynamics of the

> 1**parameters of the evolved models (with the highest subjective fitness) in the**

physical and simulated coevolution runs. The dynamics show good correspondence. However, the convergence in the physical coevolutions is somewhat less smooth than that in the simulated ones (e.g., see spikes in vℓ0 and vℓ1). In each generation of every coevolution run (physical and simulated), we computed the MAE of each model. We compared the error of the model with the highest subjective fitness with the average and lowest errors. The results are shown in Figure 4.9. For both the physical and simulated coevolution runs, the subjectively best model (green) has an error in between the lowest error (blue) and the average error (red) in the majority of generations. As we argued before (Section 3.3.1), in swarm systems, good agreement between local behaviors (e.g., controller parameters) may not guarantee similar global behaviors. For this reason, we performed 20 trials using 40 physical e-pucks, lasting 10 minutes each:

> 7**value of model parameters 1.** 5 **1** 0.5 **0**

-0.5 -1 -1.5 vℓ0 vr0 vℓ1 vr1

7value of model parameters 1. 5 1 0.5 0

-0.5 -1 1 20 40 60 generation Physical Coevolutions 80 100 vℓ0 vr0 vℓ1 vr1 -1.5 1 20 40 60 80 100 generation Simulated Coevolutions Figure 4.8: Evolutionary progress of each model parameter in 10 (a) physical and (b) simulated coevolution runs. Curves represent median values across 10 runs. Dotted black lines indicate true values.

410 trials with the original controller (Equation (3.7)), and 10 trials with a controller

obtained from the physical coevolution runs. This latter controller was constructed by taking the median values of the parameters over the 10 runs, which are: p = (−0.65, −0.99, 0.99, −0.99) . The set of initial configurations of the robots was common to both controllers. As it 1 average 0.8 subjective objective MAE 0.6 0.4 0.2 0 1 20 40 60 80 100 generation Physical Coevolutions 1 average 0.8 subjective objective MAE 0.6 0.4 0.2 0 1 20 40 60 80 100 generation Simulated Coevolutions Figure 4.9: Evolutionary progress of the MAE (defined in Equation (3.10)) of models in 10 (a) physical and (b) simulated coevolution runs. Curves represent median values across 10 runs. The red curve represents the average error of all models in a generation. The green and blue curves show, respectively, the errors of the models with the highest subjective and the highest objective fitness in a generation. was not necessary to extract the orientation of the robots, a red circular marker was attached to each robot so that its position can be extracted with higher accuracy in the offline analysis [21]. Figure 4.10(a) shows the proportion of robots in the largest cluster3 3“A

8cluster of robots is defined as a maximal connected subgraph of the graph defined by the robots’

po- 1 0.8 proportion 0.6 0.4 0.2 original controller 0 inferred controller 1 100 200 300 400 500 600 time (s) Largest Cluster Dynamics 10 original controller dispersion (×103) 8 inferred controller 6 4 2 0 1 100 200 300 400 500 600 time (s) Dispersion Dynamics Figure 4.10: Average aggregation dynamics in 10 physical trials with 40 e-puck robots executing the original controller (red) and the evolved controller (blue). In (a), the vertical axis shows the proportion of robots in the largest cluster; in (b), it shows the robots’ dispersion (see Section 3.3.1). Dotted lines in (a) and (b), respectively, represent the maximum proportion and minimum dispersion that 40 robots can achieve. over time with the original and inferred controllers. Figure 4.10(b) shows the dispersion (as defined in Section 3.3.1) of the robots over time with the two controllers. The aggregation dynamics of the original and inferred behaviors show good correspondence. sitions,

8where two robots are considered to be adjacent if another robot cannot fit between them”

[21]. 89 600 original controller 500 inferred controller time (s) 400 300 200 100 0 20 40 60 80 100 number of robots Figure 4.11: Time taken for different

8numbers of robots to aggregate into a single cluster. Each box

corresponds to 30 simulation trials with the real controller (red), and the controller obtained from the physical coevolution runs (blue).

164Figure 4. 12 shows a sequence of snapshots from a

trial with 40 e-pucks executing the evolved controller. We also performed a scalability study in simulation to compare the real and evolved controllers. We calculated the

66time taken to form a single cluster with different numbers of robots: n

∈ {10, 20, . . . , 100}. For each number of robots, we performed 30 trials with each controller. The

74results are shown in Figure 4.11. In all cases, the real controller

slightly outperforms the evolved one. However, with either controller, the robots aggre- gate in a relatively short time (< 600 s). There is a statically significant difference in aggregation times with the two controllers for n = 10, 30, 50, 70, 80, 90—but this suggests no particular pattern with respect to n. We also performed a linear least squares regres- sion on the aggregation times with the two controllers. The fits were: T = 53.2+2.2n for the real controller, and T = 64 + 2.4n for the evolved controller (n > 1). This indicates that the evolved controller does not scale much worse than the real one. A video accompanying this paper shows the evolutionary process of the models (in a particular coevolution run) both in simulation and on the physical system. Additionally, videos of all 20 post-evaluation trials with 40 e-pucks, are available in [186]. initial configuration after 20 s after 40 s after 180 s after 360 s after 420 s after 480 s after 600 s Figure 4.12: Example of a collective behavior obtained by Turing Learning. A group of 40 e-puck robots, each executing the inferred model, aggregates in a single spot. 4.5.2 Analysis of Evolved Classifiers When post-evaluating the classifiers evolved in the physical coevolution runs, we limited the number of candidate models to 100, in order to reduce the experimental time. Each candidate model was randomly chosen with uniform distribution from [−1, 1]4. Fig- ure 4.13

1shows the average decision accuracy of the best classifiers and classifier systems over the

10 coevolution runs. Similar to the results in simulation, the classifier systems obtained in the physical coevolution runs still have a high decision accuracy. The classi- 1 0.9 decision accuracy 0.8 0.7 0.6 0.5 0.4 best classifier (subjective) classifier system (subjective) best classifier (archive) classifier system (archive) best classifier (decision) 12 4 6 8 10 40 60 80 100 generation Figure 4.13: The average decision accuracy of the best classifiers and classifier systems over generations (nonlinear scale) in 10 physical coevolution runs. The error bars show standard deviations. See text for details. fier system (archive) has a higher accuracy than that of the classifier system (subjective) and best classifier (objective). However, in contrast to simulation, the decision accuracy of the best classifier (subjective) and best classifier (archive) does not drop within 100 generations. This could be due to the smaller model population size in the physical coevolution runs, which may have prevented the classifiers from getting over-specialized in a short time. To investigate whether the decision accuracy is still a good measurement of the quality of the classifiers in the physical coevolution runs, we plot a figure showing the fitness of the best classifiers corresponding to Figure 4.13. The fitness is calculated based on the 100 randomly generated models. The results are shown in Figure 4.14. Similar to the results obtained in simulation, the trend of fitness curves show good correspondence to that of the decision accuracy in Figure 4.13. The gap between the best classifier (decision) and best classifier (objective) in the physical experiments is bigger than that in simulation. One reason for this may be the limitations in motion data capture and extraction. Note that given the relatively small diameter of the e-puck in our arena, inferring its orientation is particularly challenging. 1 0.9

46fitness value 0. 8 0. 7 0. 6 0. 5 0. 4 best

classifier (subjective) best classifier (archive) best classifier (decision) best classifier (objective) 12 4 6 8 10 40 60 80 100 generation Figure 4.14:

1This plot shows the average fitness of the classifiers

over generations (non- linear scale) in 10 physical coevolution runs. The fitness is calculated based on 100 randomly generated models. See text for details. 1 0.9 1 0.8 0.8 0.7 0.6 0.6 F 0.4 0.5 0.2 0.4 0.3 0 1 0.8 0.2 1 0.6 0.8 0.4 0.6 0.1 0.2 0.2 0.4 0 0 m/n 0 f Figure 4.15: This

73plot shows the relationship between the old fitness, f , and the

new fitness, F , of the classifiers/models, when the failure ratio of individuals in the swarm, mn , changes. When m = 0 or mn → 0, F = f . When 0 < mn < 1, F is shifted, but the fitness order of the classifiers/models is not changed. See text for details. 4.6 Analysis of Sensitivity for Individual Failure In the physical experiments, we observed that some robots may get stuck or stop working because of the reasons

mentioned in Section 4.4. This could also happen in a real swarm system, for example, some individuals may stop moving or display other abnormal behaviors due to various factors such as collision. These abnormal behaviors (which are considered as a failure here) may not be recognized during the process of experiments. Therefore, we have analyzed whether abnormal behaviors (failure) of some individuals may bias the evolution and hence affect performance of our proposed coevolutionary approach. Let us assume the failure is equally likely to be present in the replica and agent in a trial. We then can prove that: Turing Learning is not biased by failure of individuals in the swarm. The mathematical proof is provided

52as follows: Proof. Let n be the total size of the

group, consisting of n − 1 agents and 1 replica. m represents the number of individuals that fail (m < n) in a trial. p and q are the prob- abilities of a classifier to make the correct judgment for models and agents respectively. Therefore, the expected fitness of the classifier without failure of any individuals in one generation, fc, is equal to p+2q . fm denotes the fitness of a model in that generation. There are two failure cases: 1) failure with the replica; 2) failure without the replica. We assume that if an individual fails during a trial, the classifier makes random judgment, that is, it has equal probability, which is 50%, of judging the individual as an agent or a model. If failure exists in a trial, the probability that failure with the replica occurs, p1, is as follows: p1 =

182n − 1 / n = m . (4 .1) (m − 1) ( ) m n

The probability that failure without the replica occurs in a trial, p2, is: 1 − p1 = n−nm. The proof is divided into two parts: one for the classifiers and the other for models. For the classifier, the new fitness in the first failure case, fc1, can be calculated as follows: fc1 = p1 · (+ 1 1 m−1 q

47·(n−m) 2 2 · n−1 + n−1)/2. (4.

2) 4.7 Summary In the second failure case, the new fitness of the classifier, fc2, is given by: fc2 = p2 · (p + 1 · m q

47·(n−m−1) 2 n − 1 + n−1)/2. (4.

3) Combining Eqs. (4.2) and (4.3) and simplifying the resulting expression, the new fitness, Fc, of the classifier when m individuals fail in a trial is: mm Fc = fc1 + fc2 = fc · (1 −) + 2n n . (4.4) In Equation(4.4), Fc is monotonic increasing in terms of fc. When fc, is equal to 0.5, Fc maintains the same value. When fc is greater than (or smaller than) 0.5, Fc is decreased (or increased) but still greater (or smaller) than 0.5. Therefore, the fitness order of all the classifiers after failure of m individuals in a trial is not changed. This means it does not affect the selection of the classifiers in the evolutionary process. The model gets a new fitness of fm1 = mn · 12 and fm2 = (n−mn)·fm in the first and second failure case, respectively. Therefore, the new fitness of the model, Fm, when m individuals fail in a trial is: mm Fm = fm1 + fm2 = fm · (1 −) + 2n n . (4.5) Comparing Eqs. (4.4) and (4.5), we can see that the fitness order of all the models is also unchanged when failure occurs in a trial. 4.7 Summary In this chapter, the Turing Learning method was validated using a physical autonomous coevolutionary system. We applied it to identify the aggregation behavior in swarms of e-puck robots. The behavior was learned successfully, and the results obtained in the physical experiments showed good correspondence to those obtained in simulation. This shows the robustness of our method with respect to noise and uncertainties in real world. Turing Learning could be beneficial in the context of science automation [187, 188], in particular to reveal the mechanisms underpinning collective animal behavior. The model obtained in the physical coevolution runs was validated using 40 e-puck robots. The global behavior of the obtained model is similar to that of the original controller. The selected classifier system over the coevolution runs still obtained a high performance, which means our classifier system could be potentially applied to detect abnormal behaviors (e.g., faulty agents in a swarm).

73In order to speed up the coevolutionary process, different from simulation, we

used two replicas instead of one. Therefore, two models can be executed simultaneously. In principle, we could further speed up the process through using more replicas in parallel. However, considering the

reliability of communication between the replicas and PC as well as the influence of replicas on the whole swarm behaviors, a limited number of replicas is recommended. We have also proved that Turing Learning is robust to failure of individuals in the mixed group. That is, even if some individuals fail during the trials, this will not bias the models obtained. 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.1 Introduction In the previous chapters, we demonstrated how Turing Learning can be used for inferring swarm behaviors through observation. This

189**is based on the** implicit **assumption that the** behavioral repertoire of **agents**

in the swarm can be fully revealed through obser- vation. From the perspective of system identification, the target system then has high observability. However, when a target system has low observability [189], the agent's behavioral repertoire may not be fully revealed through observation. Moreover, only randomly generating the sequences of inputs to the system may not be insufficient too. The machine would then need to interact with the agent to explore its hidden informa- tion. Based on this idea, in this chapter we aim to infer agent behaviors that have low observability. In particular, we extend Turing Learning (as defined in Chapter 3) with interactive capability. Observation and interaction are widely adapted by ethologists when investigating the be- havior of animals [190, 191, 192]. When investigating animals in their natural habitats, passive observation is preferable as it is difficult to change the environmental stimuli. In this case, inferring the causal relations between the animal's behavior and its environ- mental stimuli may become challenging, since the stimuli are not under the observer's control. However, when the experiments are carried out in indoor laboratories, it is possible to change/control the stimuli to interact with the animals under investigation in a meaningful way. In [192], in order to investigate the cause of the dung beetle dance, biologists designed various experiments to interact with it and learn how it adapts to the environmental changes (e.g., appearance of disturbance or obstacles). 5 Inferring Individual Behaviors Through Interactive Turing Learning In this chapter, we investigate whether a machine could automatically infer agent behav- iors (with low observability) through controlled interactions using the extended Turing Learning method. To validate this, two case studies are presented: deterministic agent behavior (Section 5.3.1) and stochastic agent behavior (Section 5.4.1). In these two case studies, the machine is

2**able to control the** agent's **environmental conditions,** which **in this**

work corresponds to the intensity of the ambient light. At the same time, it is capable of simulating the actions of the agent. The learning result is a model of the agent that captures its behavior in relation to the environmental stimulus.

10**This chapter is organized as follows. Section** 5 **.2** describes **the methodology,**

illustrating how Turing Learning is extended to have interactive capability. The deterministic and stochastic behaviors under investigation are presented as two case studies (Sections 5.3 and 5.4). Section 5.3.1 describes the deterministic behavior. Section 5.3.2 presents the simulation setup.

30**Section 5.3.** 3 **presents the results of** inferring **the**

deterministic behavior, including analysis of the evolved models, the coevolutionary fitness dynamics, analysis of the evolved classifiers, a study showing the method's sensitivity to noise, and a comparison of Turing Learning with two metric-based methods—a single-population evolutionary approach and an approach based on coevolution of inputs and models. Section 5.4.1 describes the stochastic behavior (using a state machine) for the general case. Section 5.4.2 presents the simulation setup for inferring the stochastic behavior. Sections 5.4.3 and 5.4.4 present the obtained results for the case of 2 states and 3 states, respectively. Section 5.5 summaries the chapter. 5.2 Methodology We extend Turing Learning (as described in Chapter 3) with interactive capability. In the following, we will describe the implementation that is related to the work in this chapter. 5.2.1 Models The models are represented by a set of parameters that govern the rules of the agents. The details of these parameters will be described in Sections 5.3.1 and 5.4.1. As we 5.2 Methodology

1**Figure 5.** 1: **This** diagram **shows the** structure **of the classifiers**

used in this chapter. It is a recurrent Elman neural network [176] with i input neurons, h hidden neurons, and two output neurons (O1 and O2). O1, which controls the stimulus, is fed back into the input; O2 is used for making a judgment. A bias neuron with a constant input of 1.0 is connected to each neuron of the hidden and output layers. See text for details. have argued in the previous chapters, explicit representation (i.e., evolving only the parameters) and knowing the ground truth (i.e., real parameters) makes it feasible for us to objectively gauge the quality of the models obtained. 5.2.2 Classifiers Figure 5.1 shows the structure of the classifiers. The structure is similar to the one used in Chapter 3 (see Figure 3.1). However, the classifier (neural network) has an additional output that is used to control an environmental stimulus. Moreover, the environmental stimulus is also fed into the classifier as an additional input. In the following, we will explain how the classifier works. Suppose the agent responds to the level of light intensity in the environment. We assume that the classifier can observe the agent's speed. The classifier (network) has two inputs. One

2**is the light intensity in the environment at time step t, I(t) ∈ [0, 1],** and **the other is the speed** v **(t)**

of the agent. The

2**speed of the** individual **for the classifier's input**

1**is calculated by subtracting the previous** estimated position **from the current** estimated position, **and dividing the resulting number by the time interval between two measurements. In order to make a judgment between a model and** the agent, **the classifier observes the behavior**

9**(speed) over a period of time.**

In addition, the classifier

2**is also in control of the light intensity in the** individual's environment. **At time t = 0,**

the value of the light intensity is

2**chosen randomly with a uniform distribution in the range [0, 1]. The neural network is then updated, using I(0) and** v(0). **The value of the light intensity for the next time step is obtained from the** classifier's **output neuron** O1, **and the process repeats. After having iterated through all the time steps** (a single trial), **the final value of output neuron** O2 **is used to make a** judgment: **the network decides on a model if**

O2 <

1**0.5, and on** the agent **if** O2 ≥ **0.5. The memory** (value **of** hidden neurons) of **the** classifiers **is reset**

at the end of every trial. 5

2**.2.3 Optimization Algorithm The algorithm used here is based on a (μ + λ) evolution strategy with self-adaptive mutation strengths**

[177, 109]. It is the same as the one used in Chapter 3. For the details of the implementation, see Section 3.1.1.3. 5.2.4 Fitness Calculation Suppose the population sizes for the model and classifier are M and C, respectively. The

> 1**fitness of each model is obtained by evaluating it with each of the classifiers in the** competing **population** (C **in total). For every classifier that wrongly judges the model as being the** agent, **the model's fitness increases by** C1 . **The** final **fitness** is **in [0, 1]. The fitness** of each classifier **is**

> 2**obtained by using it to evaluate (i) each model in the competing population** (M **in total) once, and (ii) the** agent L **times**

with different initial light intensities. For each

> 1**correct judgment of the model** and **the** agent, the **classifier's fitness increases by**

2·1M and 21·L , respectively. The final fitness is in [0, 1]. 5.3 Case Study One 5.3 Case Study One To validate our method, we present two case studies: one with deterministic behaviors and one with stochastic behaviors. The behaviors to be identified in this chapter were chosen to serve as a proof of concept study.

> 2**While it may loosely correspond to how some animals react to the** stimuli **in their environment, it is not intended to mimic any specific animal. In** these behaviors, **non-trivial interaction with the** agent **is critical for leading the** agent **to reveal** all **of its behavioral repertoire.**

5.3.1 Deterministic Behavior

> 2**We simulate a one-dimensional environment in** continuous **space. The simulation** ad- vances **in discrete time steps t ∈ {0,1,2,...}. The (ambient) light intensity in the** environment, **I,**

can be varied continuously between 0 and 1. The agent

> 2**distinguishes between three levels of light** intensity, **low (0 ≤ I**< IL), medium (IL ≤ **I**

≤ IH), and high (IH < I ≤ 1). The two constants, IL and IH, represent the threshold of low and high levels of the light intensity, respectively. Hereafter, the three levels

> 2**will be referred to as L, M,** and H. **If the** light intensity **is** at level **M at time t,**

the speed of the agent, s

> 2**(t) ∈ R, varies linearly with I(t) as: s(t) = k I (t) − 0.5**

, (5.1) where k is a constant. ( ) We define two constants: c1 = k (IH − 0.5) and c2 = k (IL − 0.5). The deterministic behavior under investigation is shown in Figure 5.2. The agent's behaviors for levels L and H depend on the previous levels of light intensity (i.e. the agent has memory). The two behaviors are

symmetrical to each other. Here, we will describe the behavior for level L; the behavior for level H is obtained by exchanging L with H and IL with IH in the following description. When the light intensity is at level L, the agent's default speed is k (IL − 0.5) and re- mains at that value as long as the light intensity remains at level L. If the light intensity Figure 5.2: The deterministic behavior under investigation. It shows how the agent responses to the level of light intensity (L, M and H) in its environment. Each state represents the agent's speed. See text for details. Table 5

2.1: This table shows the change of the agent's speed (shown in

Figure 5.2), for an example sequence of light levels. level

20M H L H L L L H H L L

speed k(I − 0.5) c1 c2 c1 c2 α11c2 α21c2 c1 α11c1 c2 α12c2

117L H H L L H H H M H L L

α22c2 c1 α21c1 c2 α31c2 c1 α13c1 α23c1 k(I − 0.5) c1 c2 α11c2 is at level H (for any number of time steps), and then immediately changes to level L, and remains at that level for at least one more time step, then the agent's speed decays exponentially with a rate of α1: that is, in the first time step that the light inten- sity is at level L, the agent's speed is α01k (IL − 0.5); it then changes to α11k (IL − 0.5), α12k (IL − 0.5), and so on as long as the light intensity remains at level L. The agent now registers that α1 has been activated. If another H → L → L sequence is observed, the agent's speed now decays exponentially with a rate of α2. If further H → L → L se- quences are observed, the exponential decay rate becomes and remains at α3. Note that at any time, if the agent observes a light intensity at level M, its speed is proportional to the light intensity, as shown in Equation 5.1, and it forgets all its past observations of the light

177intensity (i.e., the memory of the agent is reset). The behavior of

the agent can thus be represented by five cases; one where the agent's response to the light intensity is proportional (which occurs whenever the light intensity is at level M); one where the agent's response is constant (i.e. the agent's speed reaches the lower and upper saturation values (c1 or c2) as shown in Figure 5.2); and three where the agent's response decays exponentially with the decay rates α1, α2 and α3, respectively. Table 5

2.1 shows an example sequence of light levels, along with the corresponding speed of the

agent

9(i.e., the speed shown in

Figure 5.2). Here, IL and IH

2are set to 0.1 and 0.9 respectively. k is set to 1.25; hence, the lower and the upper saturation values of the speed are k (IL − 0.5) = −0.5 and k (IH − 0.5) = 0.5. The exponential decay rates are set to:

α1 = 0.8, α2 = 0.4, α3 =

20.2. Thus, in each case, the agent's speed decays exponentially towards zero.

Note that these values (k, α1, α2 and α3) are chosen arbitrarily and the coevolutionary algorithm is not sensitive to them. The system identification task is to learn these four parameters (k, α1, α2 and α3) of the agent. It is worth while to mention again that to learn α3, the machine needs to at least output the sequence:

> 20**H → L → L → H → L → L → H → L → L** or L → **H → H → L → H → H → L → H → H.**

Since IL and IH are set to a relatively small and large value in [0, 1] respectively, it is very unlikely to randomly generate such sequences. 5.3.2 Simulation Setup We use three setups for the Turing Learning method. The

> 2**setup, in which the classifier is in control of the light intensity in the** agent's **environment, is hereafter referred to as the**"Interactive" **setup. In order to validate the advantages of the interactive approach, we compared it against the situation where the classifier only observes the**

agent in a passive manner; that is, it does not control the light intensity in the environment.

> 2**We considered two such** setups: **in the first setup (hereafter,**"Passive 1") **the light** intensity is **randomly** chosen from the **uniform distribution in [0, 1], in every time step. In the**

> 2**2 1.5 value of model parameters 1 0.5 0 −0.5 −1 Interactive**−1.5 **Passive 1**−2 **Passive 2**

k α1 α2 α3 model parameters Figure 5.3: This

> 1**plot shows the** distributions **of the evolved models with the highest**

subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 100 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot.

> 2**second setup (hereafter,**"Passive 2"), **the** light **intensity** is **randomly**

chosen only at specific time intervals after a certain number of time steps elapsed (in this setup the number is chosen to be 10).

> 2**All other aspects of these two setups are identical to the**"Interactive" **setup.**

The population sizes of the models and classifiers are chosen to be 100, respectively. We performed 100 coevolution runs for each setup. Each coevolution run lasts 1000 generation. In one generation, each classifier conducts 100 trials on the agent. In each trial, the classifier observes the agent for

> 1**10 s at 0.1 s intervals,** that is, **a total of**

100 data points. 5.3.3 Results 5.3

> 7**.3.1 Analysis of Evolved Models**

Figure 5

1.3 shows a box plot with the distributions of the evolved models with the highest subjective fitness in the 1000th generation

over 100 coevolution runs of the three setups. 1015 Interactive Passive 1 total square error (log scale) 1010 Passive 2 105 100 10−5 10−10 1 200 400 600 800 1000 generation Figure 5.4:

1This plot shows the total square errors of the evolved model parameters

compared to those of the agent over generations. The models with the highest subjective fitness in each generation are selected. Each box corresponds to 100 coevolution runs, and the solid lines correspond to the median error. The

2passive coevolutions are able to evolve the parameters k andα1 with a reasonable accuracy; however,

they are not able to evolve α2 and α3. In the Passive 1 coevolution, the relative errors of the medians of the four evolved parameters (k, α1, α2, α3) with respect to those of the agent are 1.2%, 14.3%, 7.8 × 104%, and 2.3 × 105%, respectively. The Passive 2 coevolution leads to similarly large relative errors in the evolved values of α2 and α3. This phenomenon can be explained as follows.

9If the light intensity changes randomly

(either every time step, or every ten time steps),

2it is unlikely that the H → L→ L and/or L → H

→ H sequences will

2occur enough times, without a level of M in between, such that the classifiers can observe the effects ofα2 andα3. Therefore, the classifiers do not evolve the

ability to distinguish the

2behavior of models from the behavior of the agent with respect to these two parameters, and in turn, these parameters do not converge to their true value in the model population.

In

9contrast to the passive coevolutions, the Interactive

coevolution is

9able to evolve all the four parameters with a

good accuracy. The relative median errors are 0.024%, 0%, 0.025% and 0.15% for k, α1, α2 and α3

respectively. This implies that by the 1000th generation, the

2**classifiers have learned how to control the pattern of the light intensity in such a way that they can distinguish models from the** agent based on **the effect of any of the** four **parameters.** Therefore, in order **to**

compete for being selected in the 10 2 100 square

125**error (log scale) 10−2 10−4 10−6 10−8**

k 1 10−10 0 α 3 2 200 400 600 800 1000 generation Figure 5.5:

2**This plot shows how the square error in the individual** model **parameters changes over the** generations **in the Interactive coevolution. The**

curves cor- respond

9**to median values from 100** coevolution **runs.**

population, the models are evolved to behave like the agent in every aspect. 5.3.3.2

2**Coevolutionary Dynamics** Figure **5.** 4 **shows the dynamics of the coevolutionary** algorithms. **The**

2**horizontal axis shows the generation,** whereas **the vertical axis shows the total square error of the model parameters,** that is, **the sum of the square** errors in **the four parameters (of the**

model with the highest subjective fitness in each generation) with respect to their true values. In the

2**case of the Interactive coevolution, the** median **error starts to reduce after around the** 100th **generation, and**

keeps decreasing until the last generation where it reaches a value of 10−4. In contrast, in the

2**case of the passive** coevolutions, **not only does the** median **error not decrease, but it increases**

to a value of 108 by the 1000th generation. We now

2**analyze how the four individual** parameters **evolve during the course of the**

Interactive coevolution, which is the only fully-successful setup. The plot shown in Fig- ure 5.5

2**reveals how** the **learning proceeds in the** coevolution. **Parameter k is the first to be learnt, followed**

by α1, while parameters α2 and α3

2take a longer time to approximate the true values. This means that the classifiers first learn to distinguish models from the agent on the basis of k andα1. This ability of the classifiers drives the model population to

evolve k and α1, in order to mislead the classifiers. Eventually, the classifiers also 1

3fitness value (normalized) 0. 8 0.6 0. 4 0.

2

1average fitness of models best fitness of models 0 average fitness of classifiers best fitness of classifiers 0 200 400 600 800 1000 generation

Interactive coevolution 1

3fitness value (normalized) 0. 8 0.6 0. 4 0.

2

1average fitness of models best fitness of models 0 average fitness of classifiers best fitness of classifiers 0 200 400 600 800 1000 generation

Passive 1 coevolution Figure 5

2.6: This plot shows the subjective fitness (normalized) of the classifiers and the models in (a) the Interactive coevolution, and (b) the Passive 1 coevolution. The curves show the average fitness across 100 coevolution runs.

2learn to exploit the effects ofα2 andα3 in order to make the right judgment; thereby driving the model population to evolve these two parameters accurately. After about the 600th generation, the learning of the four parameters proceeds with approximately identical rates.

2In order to analyze why the Interactive coevolution is successful while the passive ones are not, we can look at the dynamics of the subjective fitnesses of the classifiers and the models (as defined in Section 5 .2.4) during the course of the coevolution. As both of the passive coevolutions fail to converge, we present the

analysis of fitness dynamics only for Passive 1 coevolution (the other case was found to have similar dynamics). Figure 5

2.6 shows the fitness dynamics of the Interactive and the Passive 1 coevolutions. In the case of the Interactive

coevolution (see Figure 5.6(a)), the

2average fitness of the classifiers starts off at

20.5, which means that the classifiers make judgments that are no better than random judgments. However, the classifiers

2quickly improve in fitness, which in turn causes the fitness of the models to decrease. This increases the selective pressure on the models.

After about 600 generations both the

9fitness of classifiers and models reach a

steady state, which according to Figure 5.5 corresponds to the region where the four parameters evolve with virtually identical rates. In the

9case of the Passive 1 coevolution

(see Figure 5.6(b)), the

2average fitness of the classifiers also starts off at

20.5. In the first few generations, this increases slightly, because the classifiers learn how to distinguish models from the agent on the basis of parameters k andα1. However, the models quickly adapt to this new ability of the classifiers. Now, as the classifiers are unlikely to have the opportunity to observe the effects ofα2 andα3, their average fitness returns to 0.5. This leads to a disengagement phenomenon, in which there is no more meaningful selection in the model population, therefore leading

the parameters α2 and α3 to drift, the effect of which can be seen in Figure 5.4. 5.3.3.3 Analysis of Evolved Classifiers In this section,

7we analyze the evolved classifiers in the

Interactive coevolution. The model described in Section 5.2.1 is defined by four parameters: {k, α1, α2, α3}.

1In order to evaluate the quality of the evolved classifiers we

8performed a grid search over the space of

the amount of disturbance (i.e. noise) injected into each of the four parameters. We used 11 noise magnitude settings per parameter, M ∈ {0, 0.1, . . . , 1}, with the noise being added to the parameters as follows: p′ = p(1 + U (−M, M)), (5.2) where p ∈ {k,α1,α2,α3} represents any of the four parameters, and U(−M,M) denotes a uniform distribution on the interval (−M,M).

168Note that M = 0 corresponds to no noise, whereas M = 1

means that the noisy value of the parameter can be between 0 and twice its actual value. 1 α1 0 0 k 1 1 α3 0 0 k 1 1 α3 0 0 α1 1 1

147**α3 1.0 0 0 α2 1 0.** 5 1 **α2**

0 0 α1 1 0.05 0.1 1 α2 0.01 0 0 k 1 Figure 5.7: This figure shows the landscape of U ∗ (log scale) for the overall best classi- fier over the six sub-spaces with two parameters as degrees of freedom (from 100 coevolution runs). Each axis in each plot ranges between 0 and 1, cor- responding to the minimum and maximum magnitude of noise added into each parameter respectively. See text for details. 1 αk1 αα23 judgment of the classifier 0.8 0.6 0.4 0.2 0 0 0.02 0.04 0.06 0.08 0.1 amount of parameter disturbance Figure 5.8: This plot shows the average judgment [U , see Equation 5.3] of the best evolved classifier over 100 trials when the magnitude of noise added into each parameter of the agent is within 0.1. For the sake of simplicity, we only analyzed the classifiers

1**with the highest** subjective **fitness in the** last **generation** of **the**

100 coevolutions. For each of the 100 classifiers, and value of light intensity/speed 1 0.5

2**0 −0.5** value of **light intensity/speed −1 0**

1 0.5 0 10 −0.5 −1 0 10 light intensity; L, M, H speed 20 30 40 50 time step (a) light intensity; L, M, H speed 20 30 40 50 time step (c) value of light intensity/speed 1 0.5

2**0 −0.5** value of **light intensity/speed −1 0**

1 0.5 0 10 −0.5 −1 0 10 light intensity; L, M, H speed 20 30 40 50 time step (b) light intensity; L, M, H speed 20 30 40 50 time step (d) Figure 5.9: This plot shows the light intensity sequences as output by the overall best classifier (circular points), along with the corresponding speeds of the agents (diamond-shaped points) in four trials conducted on different agents: (a) the agent; (b) a model similar to the agent, generated randomly according to Equation 5.2 with M = 0.5; (c) a model whose speed is linear to the light intensity; and (d) a static model, whose speed is always 0 (i.e. k = 0). The colors of the circular points (red, green, blue) correspond to light intensities at levels L, M and H, respectively. for each combination of noise magnitudes, we conducted 100 trials. In other words, we conducted 100 · 114 · 100 = 146, 410, 000 trials in total. In each trial, we modulated the agent's parameters according to Equation 5.2. Each trial was run for T = 100 time steps (the same setting used within the coevolutions). For each combination of magnitudes employed, the overall performance U was computed as the sum of the final judgments of the classifier in each trial,

10**divided by the** total **number of** trials, **N : N** U = Ji **/N,**

(5.3) ∑i=1 where, Ji ∈ {0, 1} is the final judgment of the classifier in trial

65**i, and N is the total number of**

trials conducted. Note that Ji = 0 and Ji = 1 imply that the classifier has judged the behavior as being that of a model and the agent, respectively. In order to find the overall best classifier from the 100 classifiers analyzed, we defined the following metric: W = [1 − U (0, 0, 0, 0)] + 1 Ω

135**(i + j + k + ℓ) U (i, j, k,**ℓ) , (5.4) **∑i,** ij2 **,k+,∑**

jℓ2∈+{0k∑,20+.1ℓ,.2.∑≠.,10} where Ω = 29282

1**is the maximum value that the** quadruple **sum can achieve (i.e. if all the** U 's **are equal to 1).**

The first term in Equation 5.4 penalizes the classifier if U < 1 when there is no noise on the parameters; they are thus undisturbed and identical to the ones of the agent. The second term penalizes the classifier if U > 0 for any non-zero noise combination, with increasing penalties being applied to higher noise magnitudes. The normalization of the second term by Ω serves to make the two terms contribute equally to W , which can take values in [0, 2]. Note that the minimum value of W = 0 can only be achieved by the perfect classifier, that is, one that outputs 1 if i = j = k = ℓ = 0 and 0 otherwise. In our case, the best classifier achieved a value of W = 3.7 · 10−3. The performance landscape of the models is

> 55-dimensional (4 model parameters plus performance measure), and cannot be visualized directly. Therefore, we considered each combination of two model parameters (42 = 6 combinations) as a sub-space, and for each point in this sub-space, we calculat (ed) the performance measure as the maximum value over the

sub-space spanned by the remaining two parameters.

> 8For instance, on the sub-space

(k, α1), the performance measure U ∗ (k, α1) was calculated as: U ∗ (k, α1) = max U (k, α1, α2, α3) . (5.5) α2 ,α3 Note that for all the points except (0, 0, 0, 0), Equation 5.5

> 183corresponds to the worst -case scenario for the classifiers, because the

value of U for the ideal classifier at these points is 0. For the point (0, 0, 0, 0), the value of U for the ideal classifier is 1. Figure 5.7 shows the landscape of U∗ (log scale) for the overall best classifier over the six sub-spaces. When interacting with the agent (i.e. i = j = k = ℓ = 0), the output of the classifier is always 1. This corresponds to the point U∗(0,0) in the six sub-spaces of Figure 5.7 (here only the maximum is shown). When interacting with the models (i.e. when the parameters of the agent are perturbed), for any combination of noise magnitudes, the average output of the classifier is below 0.05 for the six sub-spaces. In order to further analyze how sensitive the overall

> 1best classifier is when the parameters of the agent are

only slightly perturbed, we set the maximum magnitude of noise added to each parameter to 0.1, and used a resolution of 0.01. In this evaluation, we only injected noise into one parameter at a time. For each noise magnitude, the classifier was evaluated in 100 trials (with different initial light intensities, and randomly-generated noise), and the average performance measure was computed according to Equation 5.3. Figure 5.8 shows the average judgment [U, see Equation 5.3] of the best classifier, which was evaluated in 100 trials. With increasing noise magnitudes, the average judgment of the classifier approaches zero, which means that the classifier can identify the agents as models more consistently. The classifier has different sensitivities to the four parameters; the effects of noise on α2 and α3 on its judgment are higher than those of k and α1. For instance, in the case of α2, even with M = 0.01, the classifier correctly judges the behavior as being that of a model in 98% of the trials. As we have seen, k and α1 are the first two parameters to be identified in the coevolution. α2 and α3 are addressed in the later generations of the coevolutionary process, and it seems that the classifier is more sensitive to these two parameters. We now investigate how the overall best classifier interacts with the agents. We con- ducted four trials with four different agents: (a) the agent; (b) a model similar to the agent, generated randomly according to Equation 5.2 with M = 0.5; (c) a model whose speed is linear to the light intensity without exponential decay (i.e. k = 1.25, α1 = α2 = α3 = 1); and (d) a static model, whose speed is always 0 (i.e. k = 0). In each trial, the initial light intensity was set to 0.25. The trials lasted for 100 time steps. Figure 5.9 shows the sequences of

> 2light intensity output by the classifier, along with the speed of the agents in the

four trials, for the first 50 time steps (we observed that the last 50 time steps constitute a repetition of the first 50). As we can see, the classifier outputs different sequences of light intensity in order to interact with

the different agents. The greater the difference between a model and the agent, the more varied is the sequence of light intensities that the classifier outputs when interacting with it as compared to the one it outputs when interacting with the agent. For the agent, the classifier repeatedly outputs a H → L → L sequence, in order to fully reveal the behavior (this is analyzed in detail in the following paragraph). For the model that is similar to the agent (see Figure 5.9(b)), the sequence of light intensities is similar to the one produced for the agent, but it is not identical. Interestingly, for the model without exponential decay (see Figure 5.9(c)), the classifier produces the H → L → L sequence even more often than it does for the agent, but it does not set the light intensity to level M. For the static model (see Figure 5.9(d)), the classifier never outputs a H → L → L sequence; instead, it outputs a sequence that alternates between M and L. We now focus on analyzing the behavior of the agent in Figure 5.9(a). Initially, the classifier outputs the sequence

> 118**L → H → L → L → H → L → L → H → L → L → M**

, which means that by the 12th

> 2**time step, it has already observed** the effects of **all four parameters.** Interestingly, **the** classifier **then**

(essentially) repeats this sequence and thus

> 9**observes the effect of all** the **parameters for**

another seven times (four repetitions occur in the last 50 time steps, which are not shown in Figure 5.9(a)). We conjecture that the repetitions

> 2**make the classifier more robust in** determining **whether the behavior** under observation **is that of a model or the**

agent. These results suggest that the classifier succeeds in controlling the level of ambient light in a way that helps distinguish between the agent and the models, revealing all aspects of the underlying behavior. Note that the classifiers are not provided with any prior information about the agent, including its structure. Thus, all knowledge is gained through an evolutionary process that is driven by the accuracy of their judgments. 5.3.3.4 Noise Study We now consider the situation where, during the coevolutionary process, noise is injected into the agent's behavior, and the agent's positions as measured by the system. This makes the overall setup more realistic as the locomotion of agents and any tracking system will be affected by noise and measurement error. Since the

> 2**passive coevolutions fail even in the noiseless case,**

we consider only the Interactive coevolution for the sake of simplicity.

> 9**We performed 100 coevolutionary runs** with **the**

following settings. The light intensity perceived by the agent at time t

> 2**is obtained by multiplying the** actual **intensity by a** random **number generated uniformly in (0.95, 1.05), and capping the perceived intensity to 1 if it exceeds this value. Noise is also applied to the speed of the**

> 1**value of model parameters**−0.5 −1 **−1.** 5 **2 1.** 5 **1**

0.5 0 −2 k α1 α2 model parameters α3 Figure 5.10: This

1plot shows the distributions of the evolved models with the highest

subjective fitness in the 1000th generation of the Interactive coevolution with noise (for a comparison to the case without noise, see Figure 5.3). The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. agent

2by multiplying the original speed with a random number generated uniformly in (0.95, 1.05).

Noise on the estimated position

2on the agent is applied by adding a random number generated from a normal distribution: N (0, 0.005).

Figure 5.10

1shows a box plot with the distributions of the evolved models with the highest subjective fitness in the 1000th generation

of the Interactive coevolution with noise. The

2effect of the noise is to widen the distribution of the evolved parameters across the 100 coevolutionary runs; however, the median values of the evolved parameters are still very close to the true values. Interestingly, the Interactive coevolution does not seem to learnα2 andα3 significantly worse than it does learn k and

α1. 5.3.3.5 Using a Single-Population

2Evolutionary Algorithm In order to compare Turing Learning against a more traditional approach, we used a simple evolution where

a single population of models evolves. We call this method SP- EA. As there are now no

2classifiers, an interactive approach is not possible, and thus we conducted 100 evolutionary runs for the Passive 1 and Passive 2 methods of changing the

22 1.5 value of model parameters 1 0.5 0 −0.5 −1 −1.5 −2 Passive 1 Passive 2

k α1 model parameters α2 α3 Figure 5.11: This plots

1shows the distributions of the evolved models with the highest

fitness in the 100000th generation in the simple evolutions with a single population. Each box corresponds to 100 evolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. Note that in order to zoom in on the relevant range, some boxes and outliers are omitted from the plot. Table

5.2: This table shows a comparison of all approaches. The numbers show the relative errors of the evolved parameters (median values over 100 runs) with respect to the parameters of the agent (in absolute percentage). TL: Turing Learning. k α1 α2 α3 Interactive (TL) 0.024 0 0.025 0.15 Passive 1 (TL) 1.2 14.3 7.8 × 104 2.3 × 105 Passive 2 (TL) 0.7 5.74 1.3 × 104 3.3 × 105 Passive 1 (SP-EA) 0 0 1.2 48.7 Passive 2 (SP-EA) 0 0 9.8 48.5 CEA-IM 4.0 × 10−5 4.9 × 10−5 1.1 × 10−3 2.5 × 10−4

2**light intensity in the** agent's **environment. The structure of the evolution is**

identical

2**to the sub-algorithms used in the coevolution,** except for **the**

fitness evaluation step.

2**Now, in each generation, 100 experiments are performed on the** agent **using 100 randomly generated intensity patterns. The 100 intensity patterns are used to evaluate a model 100 times. The average square error** between **the model's and the** agent's **speed sequences is used as the model's fitness.**

Each evolutionary run lasts 100, 000 generations. In other words, the number and duration of experiments on the agent is kept the same as that

1**value of model parameters 2 1 0 -1 -2**

k α1 α2 α3 model parameters Figure 5.12: This plots

1**shows the** distributions **of the evolved models with the highest**

fitness in the 100000th generation in the simple evolutions with a single population. Each box corresponds to 100 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e. those of the agent). From top to bottom, these are 1.25, 0.8, 0.4, 0.2, respectively. in the coevolutionary approach, as outlined in Section 5.3.2. Figure 5.11 reveals that the evolution is able to identify parameters k, α1, α2, but not α3. Note that, apart from the single-population evolution not being able to consistently identify the parameter α3, they also rely on a pre-defined metric for their operation; in this case, computed as the square error between the model's and the agent's speed sequences. 5.3.3.6 Coevolution of Inputs and Models In this section, we compare Turing Learning with another coevolutionary system iden- tification method, in which inputs and models competitively coevolve. In particular, we use the classifiers to generate sequences of inputs (in this case, light intensity). The models are optimized through minimizing the square error of speed between the agent and models, given the same sequence of inputs generated by the classifiers. The clas- H/p1 L/p1 H/p1 L 1 2 3 4 H/p2 L/p2 H/p2 Figure 5.13: A state machine which represents the stochastic behavior of the agent under investigation. The initial state is 1. p1, p2 are probabilities. The ambient light intensity that the agent responds to has two levels: H and L. See text for details. sifiers compete with the models through generating inputs that cause the maximum disagreement between the model's prediction and the agent's output, which is similar to the concept in [14]. Note that in this case the classifiers are only used for generating the inputs and they do not make judgments. We call this method CEA-IM. Figure 5.12 shows the results of CEA-IM. As we can see, the model parameters are also identified with a high accuracy. In other words, there is no 'true' interaction between the agent and classifiers during the experiments in Turing Learning when investigating the deter- ministic behavior. The classifiers only need to learn how to generate a fixed sequence of inputs to extract all the information from the agent. For a comparison of all approaches, see Table 5.2. In order to further validate and highlight the benefit of Turing Learning, we investigate the stochastic behaviors in the following section. 5.4 Case Study Two 5.4.1 Stochastic Behavior Stochastic behaviors are widely observed in the animal kingdom. Given the same stimuli, the animal may behave differently. In order to make the animal under investigation reveal all its behavioral repertoire, ethologists sometimes need to interact with the animals in real time and change the stimuli dynamically according to the animal's response. Based on this motivation, in this section we apply Turing Learning to infer stochastic behaviors H/ L 1

202 L H/ 2 states H/ L/ L 1 2 3 H H/

L/ 3 states Figure 5.14: The stochastic behaviors with (a) 2 states and (b) 3 states under invesitga- tion. p1 and p2 are probabilities. See text for details and investigate whether a machine could interact with the agent through dynamically changing the stimulus rather than applying a fixed sequence. We will describe the stochastic behavior for the general case. It is represented by a state machine, shown in Figure 5.13. Suppose the agent has n states. The agent's behavior depends on the level of the light intensity in the environment and its current state (i.e., the agent has memory). For the initial state 1, if the light intensity is low (L), the agent stays in state 1; if the light intensity is high (H), the agent moves forward to state 2 with probability p1. For states i = 2, 4, 6, · · · , 2 n−21 , if light intensity is L, the agent moves to state (i + 1) with probability p1; if th └e ligh┘ t intensity is H, the agent moves to state (i − 1) with probability p2. The behavior for states i = 3, 5, 7, · · · , 2 n2 − 1 is obtained by exchanging with in the above description. When the age └nt┘ is in L H state n, its behavior is similar to that in states i = 2, 3, 4, n − 1. The only difference is that the agent can not move to a higher state. Figures 5.14(a) and 5.14(b) show the agent behavior with 2 and 3 states, respectively. These are the two behaviors to be investigated in the thesis. For the agent's behavior, p1

173is set to a low value and p2 is set to

a high value. As a consequence of this choice, the agent has a low chance of moving forward to a state with a higher number and thus the higher state has lower observability. The classifiers need to learn how to interact with the agent to infer all its behavioral repertoire. For a proof of concept study, we assume that the agent moves in one-dimensional space and moves at a constant but different speed in each state. Suppose the agent is initially static (zero speed) and its initial state is known. The system identification task is then to identify the parameters of the other states (i.e., speed values v1 and v2 shown in Figure 5.14) and the two probabilities, p1 and p2. The speed of the agent in each state is chosen arbitrarily. p1 and p2 are chosen to have a value of 0.1 and 1.0. As explained, this makes the higher states harder to be observed. In order to make the agent stay in a higher state, the classifiers need to capture the mo- ment when the agent reaches that state, and toggles/switches the level of light intensity immediately. This makes it easier for Turing Learning to learn the behavior as the state can be observed for a sufficient time. If the classifiers fail to do that, the agent would move immediately back to the previous state. For the 2-state machine shown in Figure 5.14(a), v1 is selected to be 0.5; for the 3-state machine shown in Figure 5.14(b), v1 and v2 are selected to be 0.5 and 1.0, respectively. Therefore, the parameters to be identified for the 2-state and 3-state agent behavior are: q1 = (v, p1, p2) = (0.5, 0.1, 1.0). (5.6) q2 = (v1, v2, p1, p2) = (0.5, 1.0, 0.1, 1.0). (5.7) 5.4.2 Simulation Setup To evaluate Turing Learning, we still used three setups: "Interactive", "Passive 1" and "Passive 2", as discussed in Section 5.3.2. We also compared Turing Learning with the two metric-based methods (SP-EA and CEA-IM) described in Sections 5.3.3.5 and 5.3.3.6, respectively. For each setup, we added a certain amount of noise into the 119 measurement of speed. This is realized by multiplying the agent's real speed with a random value in [0.95, 1.05] in each time step. 5.4.3 Results: Two States In

35this section, we present the results of inferring the

2-state agent behavior shown in Figure 5.14(a), using the setups in Section 5.4.2. 5.4.3.1 Analysis of Evolved Models Figure 5.15

1shows a box plot with the parameters of the evolved models with the highest subjective fitness in the 1000th generation

for (a) Turing Learning, (b) SP-EA, and (c) CEA-IM. Using Turing Learning, the system identified all parameters of the agent with good accuracy. For the other two metric-based methods, all the three parameters are not learned well. Instead, the three evolved parameters converge into three different values: v1 → 0.0, p1 → 1.0, p2 → 0.0. The failure of SP-EA and CEA-IM can be explained as follows. As the behavior is stochastic, given the same sequence of inputs the agent would probably exhibit different behaviors. Therefore, quantitatively measuring the difference between the models and agent (e.g., using square error) would not lead the model parameters to converge into their true values. For all methods, the means (standard deviations) of the AEs of each parameter of the evolved model

1with the highest fitness in the final generation over 30 coevolution runs

are shown in Table 5.3. Clearly, Turing Learning infers the stochastic behavior significantly better than the two metric- based methods. There is no significant difference among “Interactive”, “passive 1” and “passive 2” setups of Turing Learning in terms of AEs of the evolved parameters. In order to show the advantage of “Interactive” setup of Turing Learning, we investigate the convergence of model parameters during the evolutionary process. Figure 5.16

1shows the convergence of the model parameters over generations for the

three setups of Turing Learning. As we can see, the evolved model parameters in the “Interactive” setup converge much faster than those in the two passive setups. For the “Interactive” setup, after about 100 generations, all the three parameters converge into their true values. For

1value of model parameters 2 1 0 -1 -2

Interactive Passive 1 Passive 2 v1 p1 p2 model parameters Turing Learning

1value of model parameters 2 1 0 -1 -2

v1 p1 p2 model parameters SP-EA

1value of model parameters 2 1 0 -1 -2

Passive 1 Passive 2 v1 model parameters p1 p2 CEA-IM Figure 5.15: This

1plot shows the distributions of the evolved models with the highest

subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 30 coevolution runs. The dotted lines correspond to the values of the three parameters that the system is expected to learn (i.e., those of the agent). See text for details. the “Passive 1”, all the parameters converge after about 200 generations, while for the “Passive 2” it takes longer time. In terms of v1, there is a much smaller disturbance in the “Interactive” setup than that in the other two setups.

1value of model parameters 2 1 0 -1

v1 p1 -2 p 2 1 200 400 600 800 1000 generation Interactive

1value of model parameters 2 1 0 -1

v1 p1 -2 p 2 1 200 400 600 800 1000 generation Passive 1

1value of model parameters 2 1 0 -1 -2

v1 p1 p 2 1 200 400 600 800 1000 generation Passive 2 Figure 5.16: Evolutionary process of the evolved model parameters for (a) “Interactive”, (b) “Passive 1” and (c) “Passive 2” setups of the metric-free method when learning the 2-state agent behavior. Curves represent mean values across 30 coevolution runs. Dotted black lines indicate true values. 5.4.3.2 Analysis of Evolved Classifiers In order to investigate why the “Interactive” setup of Turing Learning learns the agent behavior faster than the two passive setups, we post-evaluated how the classifiers interact with the agent during a trial. Table 5.3: This table shows a comparison of all approaches for learning the 2-state stochastic behavior shown in Figure 5.14(a). The values show

means of AEs (defined in Equation 3.9) of each evolved model parameter with respect to that of the agent. TL: Turing Learning. v1 p1 p2 Interactive (TL) 0.003 0.03 1.6 × 10−5 Passive 1 (TL) 0.01 0.03 1.0 × 10−5 Passive 2 (TL) 0.02 0.02 1.0 × 10−5 Passive 1 (SP-EA) 0.47 0.9 1.0 Passive 2 (SP-EA) 0.47 0.9 1.0 CEA-IM 0.47 0.9 1.0 Figure 5.17 shows how the classifier with the highest subjective fitness in different gen- erations (of a particular coevolution run) interact with the agent in a trial. A similar phenomenon could be observed in other coevolution runs. As shown in the top left of Figure 5.17, the classifier outputs H level. It waits until the agent 'jumps' from state 1 to state 2 (a stochastic process), and then immediately switches the light intensity from H to L level in order to make the agent stay in state 2 as long as possible. Therefore, the agent's behavior in state 2 (hidden information) can be observed longer and inferred efficiently through the process of evolution. Note that the classifier has learned a good strategy and exhibited such 'intelligent' behavior at the very beginning of the coevolu- tion run (before 50 generations in this case). After 500 generations, the classifier slightly changed the strategy. Instead of always outputting H level, it keeps switching between H and L level until it observes the agent 'jumps' from state 1 to state 2, and after that it immediately switches the light intensity from H to L level. This is unlikely to happen when generating random sequences of inputs. 5.4.4 Results: Three States In order to further demonstrate the advantage of "Interactive" setup of Turing Learning, this section discusses the results of inferring the 3-state stochastic agent behavior shown in Figure 5.14(b). speed/light −0.5 0.0 0.5 1.0 1.5 −1.5 speed/light −0.5 0.0 0.5 1.0 1.5 −1.5 generation 50 ● ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●● ●●●●● ●●●●●●●●●●●●●●●●●● ● ● ● speed light 0 20 40 60 80 100 time step generation 500 ●●●●●●● ●●●●●● ● ● ●●●●●● ●● ● ● ● ● ●●●●● ●●● ● ● ● ● ● ● ● ●● ● ● ● speed light 0 20 40 60 80 100 time step speed/light speed/light −0.5 0.0 0.5 1.0 1.5 −1.5 −0.5 0.0 0.5 1.0 1.5 −1.5 generation 100 ●●●●● ● ● ● ● ●●●●●●●●●●●●●●●●●●●●● ● ● ● ● ● ● speed light 0 20 40 60 80 100 time step generation 1000 ● ●●●●●●●●●●● ●●●●●●●● ●●●●● ●●●●●●●●●●●●● ●●●● ● ●●● ● ● ● ● speed light 0 20 40 60 80 100 time step

7**Figure 5.** 17: **This plot shows** an example **of** how **the**

classifier

7**with the highest subjective fitness in**

different generations (of a particular coevolution run) dynamically change the light intensity to interact with the 2-state agent during a trial. 5.4.4.1 Analysis of Evolved Models Figure 5.18

2**shows the** distribution **of the evolved** models **in the** 1000th **generation**

for all setups. The "Interactive" setup of Turing Learning is the only one that infers all the parameters of the agent with good accuracy. For the two setups using pre-defined metrics (SP-EA and CEA-IM), all the parameters are not well learned. Table 5.4 compares the accuracy of each parameter of the evolved model

1**with the highest fitness in the** final **generation over 30 coevolution runs** using **all**

approaches. Figure 5.19 shows the evolutionary process of the models for the three setups of Turing Learning. As we can see, all the model parameters in the "Interactive" setup converged to their true value smoothly within about 200 generations. For the two passive setups,

1**value of model parameters 2 1 0 -1 -2**

Interactive Passive 1 Passive 2 v1 v2 p1 p2 model parameters Turing Learning

1**value of model parameters 2 1 0 -1 -2**

v1 v2 p1 p2 model parameters SP-EA

1value of model parameters 2 1 0 -1 -2

Passive 1 Passive 2 v1 model parameters v2 p1 p2 CEA-IM Figure 5.18: This

1plot shows the distributions of the evolved models with the highest

subjective fitness in the 1000th generation in the coevolutions. Each box corresponds to 30 coevolution runs. The dotted lines correspond to the values of the four parameters that the system is expected to learn (i.e., those of the agent). See text for details. v1, p1 and p2 are well learned, but they still take

2longer time to converge to their true values than those of the

"Interactive" setup. There is a dramatic disturbance during the evolutionary process of v2 for the passive setups, and this parameter is not well learned.

1value of model parameters 2 1 0 -1 -2

v1 v2 p1 p 2 1 200 400 600 800 1000 generation Interactive

1value of model parameters 2 1 0 -1 -2

v1 v2 p1 p 2 1 200 400 600 800 generation 1000 Passive 1 value of model parameters 4 3 2 1 0 -1 -2 -3 -4 v1 vp21 p 2 1 200 400 600 800 1000 generation Passive 2 Figure 5.19: Evolutionary process of the evolved model parameters for (a) "Interactive", (b) "Passive 1" and (c) "Passive 2" setups of the metric-free method when learning the 3-state agent behavior. Curves represent mean values across 30 coevolution runs. Dotted black lines indicate true values. 5.4.4.2 Analysis of Evolved Classifiers Figure 5.20 shows an example of how the classifier with the highest subjective fitness in different generations (of a particular coevolution run) evolved to interact with the agent. Table 5.4: This table shows a comparison of all approaches for inferring the 3-state agent behavior shown in Figure 5.14(b). The values show means of AEs (defined in Equation 3.9) of each evolved model parameter with respect to that of the agent. TL: Turing Learning. speed/light speed/light v1 v2 p1 p2 Interactive (TL) 0.005 0.03 0.02 2.0×10^{-6} Passive 1 (TL) 0.02 0.23 0.05 0.03 Passive 2 (TL) 0.02 0.47 0.08 0.13 Passive 1 (SP-EA) 0.47 0.97 0.9 1.0 Passive 2 (SP-EA) 0.47 0.98 0.9 1.0 CEA-IM 0.46 0.96 0.67 0.89 −0.5 0.0 0.5 1.0 1.5 −1.5 −0.5 0.0 0.5 1.0 1.5 −1.5 generation 100 ● ● ● ● ● ● ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●● ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●● ●●●●●●●● ● ● ● ● ● ● speed light 0 20 40 60 80 100 time step generation 500 ● ●●●●●●●●●●●●●● ●●●●●●● ●●● ●●●● ●●● ● ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●● ●●●●●●●● ● ●●●●●●●●●● ● ● ● ● ● ●●● ● ● ●●● ● ● speed light 0 20 40 60 80 100 time step speed/light −0.5 0.0 0.5 1.0 1.5 −1.5 speed/light −0.5 0.0 0.5 1.0 1.5 −1.5 generation 200 ● ●● ● ● ● ● ● ● ● ● ● ● ● speed light 0 20 40 60 80 100 time step generation 1000 ●● ●● ●●●●●●●●●●●●●●●●●●●● ● ●●●●●●● ● ● ● ● ●● speed light 0 20 40 60 80 100 time step

7Figure 5. 20: This plot shows an example of how the

classifier

7with the highest subjective fitness in

different generations (of a particular coevolution run) dynamically change the light intensity to interact with the 3-state agent during a trial. In other coevolution runs, we observed a similar phenomenon. As shown in Figure 5.20, the strategy learned by the classifiers shown in 3 of the 4 sub-figures (corresponding to the 100th, 200th and 1000th generations) is as follows. The classifier outputs H first, and once the agent moves forward from state 1 to state 2, the classifier switches the light intensity into level L and keeps it in that level. As long as the agent moves forward from state 2 to state 3, the classifier switches the light intensity from L to H and keeps the light intensity in that level. Note that the 'best' classifier sometimes lost its ability to

interact with the agent in the 500th generation as shown in bottom left of Figure 5.20. However, this does not influence the learning process, as long as there are still some other classifiers in the population obtain this interactive ability. 5.5 Summary In this chapter, we extend Turing Learning with interactive ability to autonomously learn the behavior of an agent.

> 2**We have shown that, by allowing classifiers to control the** stimulus **in the** agent's **environment, the system is able to correctly identify the parameters of relatively complex**

behaviors. The advantage of Turing Learning with interaction is validated using two case studies: stochastic and deterministic behaviors of an agent. In both case studies, the results show that learning through interaction can infer the agent behaviors better or faster than only through passive observation. When inferring the deterministic behavior, the classifiers have learned to generate a complex but static sequence of inputs to extract all the hidden information from the agent, which facilitates the learning process. However, generating such sequences at random is unlikely. This makes the SP-EA (in which the inputs are generated ran- domly) fail to infer all the parameters of the agent. However, by coevolving inputs and models (CEA-IM), the system still obtained very good model accuracy in terms of all parameters. When inferring the stochastic behavior, the advantage of Turing Learning with inter- action becomes obvious. It performs significantly better than the two metric-based methods (SP-EA and CEA-IM) in inferring the behaviors. The classifiers learned to dynamically interact with the agent, which made it possible to infer all the agent's pa- rameters in an efficient way. In a sense, the machine, using Turing Learing, exhibits an 'intelligent' behavior known from humans: it adapts to interact with a system in order 5.5 Summary to learn faster what the system does. This property of Turing Learning could pave the way to further development of machine intelligence. 6 Conclusion 6.1 Summary of Findings This thesis presented a novel system identification method—Turing Learning—for infer- ring agent behavior.

> 4**Turing Learning does not** rely on **predefined metrics** for measuring **the difference between** the agents **and**

models. Instead, it uses coevolution to simulta- neously generate models and classifiers, which substitute the metrics. The classifiers are rewarded for distinguishing between agents and models. The models are rewarded for making classifiers judge them as agents. In order to 'trick' the classifiers to judge them as agents, the models need to mimic their behaviors. The merits of Turing Learning were demonstrated through successfully inferring various agent behaviors ranging from swarm behaviors to deterministic/stochastic behaviors of single agents. When inferring an unknown swarm behavior, it is challenging to define meaningful metrics that quantitatively measure the difference (e.g., in motion) between the models and agents. Due to the numerous

> 114**interactions among agents and between agents and the environment, the** motion **of** each agent in **the**

swarm is stochastic. We conducted a case study (with both simulated and physical robots) showing that observing the swarm is sufficient to infer the behavioral rules of the agents. The evolved models showed good correspondence to the original agents in terms of individual behaviors (parameters) and global behaviors. The evolved classifiers performed collectively well and could potentially be used for detecting abnormal behaviors (e.g., faulty agents) in the swarm. It was also shown that swarm

> 4**behaviors can be directly inferred from the motion of a single agent in the group,**

as long as the group size is sufficiently large. This

> 4**may have significant implications for the study of animal collectives,**

as in practice it may be difficult to track a large number of animals in a group. 6 Conclusion We extended the

ability of the classifiers in Turing Learning, so that they can interact with the agent through controlling the environmental stimulus that the agent responds to. The interactive learning approach proved to be superior to passive learning (i.e., learning where the agents are only observed)

> 170**in terms of** model **convergence rate and** model **accuracy,** especially when **the**

agent behavior under investigation had low observ- ability. We presented two further case studies where Turing Learning needs to infer, respectively, deterministic and stochastic behaviors of single agents. In the case study about inferring deterministic behaviors, the results showed that it is possible to infer the behaviors through coevolving a fixed sequence of inputs (in lieu of classifiers) and models. In this case, the interaction between the classifiers and agent was of limited 'intelligence' as they only needed to output a fixed sequence no matter what the agent's observed behavior was. In the case study about inferring stochastic behaviors, it was shown that through actively interacting with the agent during the experimental trial, the classifiers could 'intelligently' change the stimulus in response to the agent's observed behavior, and thereby extract the hidden information from the agent. This intelligent behavior was also observed in humans when scientists tried to investigate an animal's behavior in response to stimuli [121]. As the agent's behavior under investigation is stochastic, it is not feasible to evolve a fixed sequence of inputs to extract all the agent's behavioral repertoire. Given the same input, the agent would probably behave differ- ently each time and this makes it hard to optimize the models using predefined metrics. We compared the results obtained by Turing Learning and two metric-based system identification methods, and showed that Turing Learning learned the agent behavior significantly better than the other two methods. This highlights the benefits of Turing Learning in inferring complex agent behaviors. 6.2 Future work In spite of the encouraging results obtained when applying Turing Learning to infer agent behaviors, we do not claim that this method can be directly used for modeling animal behaviors. There are still questions to be discussed before conducting experiments on animals. Some future work is provided as follows. • In the thesis, the models were represented by a set of parameters that govern 6.2 Future work behavioral rules of the agents. As was argued before, this makes it feasible to objectively gauge the quality of the models through comparing them with the ground truth. In the future, we will try to evolve the structure of the models as well (e.g., using genetic programming or artificial neural networks). • The swarm behaviors investigated in this thesis were deterministic in terms of the individual's behavioral rules (yet, the motion of the agents was stochastic, for example, due to the noise on their actuators). In the future, we could apply Turing Learning to learn swarm behaviors produced by stochastic rules. • Turing Learning could be applied to learn more complex behaviors (for example, when the agents have more states or rules). When the behaviors become more complex, instead of analyzing only the motion of individual agents, more informa- tion (such as number of the agent's neighbors or its internal states) may need to be provided to the classifiers. • In principle, Turing Learning is applicable to infer human behavior. It could evolve models that aim to pass the Turing Test [174], at least with regards to some specific subset of human behavior. In this case, the

> 9**classifiers could** act **as Reverse Turing Tests,**

which could be applied in situations where a machine needs to distinguish human agents from artificial ones. This is done, for example, by the "Completely Automated Public Turing test to tell Computers and Humans Apart"

> 2**system (CAPTCHA)** [193], which **is widely used** for **internet security**

purposes. For example, an exciting application of Turing Learning is reverse engineering the hand writing of human beings. Given a collection of signatures from a human being, we could coevolve models which are computer programs that automatically generate signatures and classifiers (e.g., neural networks) that distinguish between signatures from the human and those generated by models. Bibliography [1] J. P. Grotzinger, "Habitability, taphonomy, and the search for organic carbon on mars,"

> 193**Science, vol. 343, no. 6169, pp. 386–387,**

2014. [2]

> 58**R. P. Hertzberg and A. J. Pope,**"High-throughput screening: new technology for

the 21st century," **Current Opinion in Chemical Biology, vol. 4, no. 4, pp. 445**

– 451, 2000. [3]

1**J. Bolhuis and L. Giraldeau, The behavior of animals: mechanisms, function, and evolution.** USA: **Wiley-** Blackwell, **2004.** [4] W. **J.**

Sutherland, "The importance of behavioural studies in conservation biology,"

159**Animal Behaviour, vol. 56, no. 4, pp. 801–809,**

1998. [5]

50**D. Floreano and C. Mattiussi, Bio-Inspired Artificial Intelligence: Theories, Meth- ods, and Technologies. Cambridge, MA: MIT Press, 2008.**

[6]

66**J.-A. Meyer and A. Guillot,**"Biologically inspired robots," **in Springer**

89**Handbook of** Robot., **ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg,** Germany: **Springer,** 2008, **pp.**

1395–1422. [7] R.

13**King, J. Rowland, S. G. Oliver,** and M. Young, "The automation of science," **Science, vol. 324, no. 5923, pp. 85–89, 2009.** [8] J. Evans **and**

17**A. Rzhetsky,**"Machine science," **Science, vol. 329, no. 5990, pp. 399–400, 2010.**

[9] D. Waltz and B. G. Buchanan, "Automating science," Sci., vol. 324, no. 5923, pp. 43–44, 2009. Bibliography [10]

107**L. Ljung,**"Perspectives on system identification," Annu. **Reviews in Control, vol. 34, no. 1, pp. 1–12, 2010.**

[11] S.

67**A. Billings, Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains.** Hoboken, NJ, USA: **Wiley, 2013.**

[12] S. M. Henson and J. L. Hayward, "The mathematics of animal behavior: An interdisciplinary dialogue," Notices of the AMS, vol. 57, no. 10, pp. 1248–1258, 2010. [13]

157**E. Bonabeau,**"Agent-based modeling: Methods and techniques for simulating human systems," **PNAS, vol. 99,** no. 10, **pp. 7280–7287, 2002.**

80**[14] J. Bongard and H. Lipson,**"Nonlinear system identification using coevolution of models and tests," **IEEE Trans. Evol.** Computation, **vol. 9, no. 4, pp. 361–384,**

2005. [15] ——, "Automated reverse engineering of nonlinear dynamical systems," PNAS, vol. 104, no. 24, pp. 9943–9948, 2007. [16]

51**G. D. Ruxton and G. Beauchamp,**"The application of genetic algorithms in be- havioural ecology, illustrated with a model of anti-predator vigilance," **Journal of Theoretical Biology, vol. 250, no. 3, pp. 435–448, 2008.** [17] S. Camazine, **J.-**

L. Deneubourg, N. R. Franks,

97**et al., Self-Organization in** Biolog- ical **Systems.** Princeton, **NJ: Princeton University Press, 2001. [18] D.**

Helbing and A. Johansson, "Pedestrian, crowd and evacuation dynamics," in Extreme Environmental Events, R. A. Meyers, Ed. Springer, 2011, pp. 697–716. [19] J. Harvey, K. Merrick, and H. A. Abbass, "Application of chaos measures to a simplified boids flocking model,"

209**Swarm** Intell., vol. **9,** no. 1, **pp. 23–41,**

2015. [20] W. S, B. S, F. R, et al., "Modeling collective animal behavior with a cognitive perspective: a methodological framework," PLoS ONE, vol. 7, no. 6, 2012, e38588. [21]

5**M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß,**"Self-organized aggregation without computation," **The** Int. J. **of** Robot. **Research, vol. 33, no. 8, pp. 1145–1161, 2014.**

Bibliography [22] ——, "Clustering objects with robots that do not compute," in Proc. 2014 Int. Conf. Autonomous Agents and Multi-Agent Syst., IFAAMAS Press, Paris, France, 2014, pp. 421–428. [23] H. Schildt,

69**Artificial intelligence using C. New York,** NY, USA: **McGraw-Hill,**

1987. [24] E. Charniak,

69**Introduction to artificial intelligence. Reading, MA,** USA: **Addison- Wesley,**

1985. [25]

29**D. B. Fogel, Evolutionary computation: toward a new philosophy of machine**

in- telligence. Street Hoboken, NJ, USA: Wiley-IEEE Press, 1995. [26] M. L.

148**Minsky,**"Logical versus analogical or symbolic versus connectionist or neat versus scruffy," **AI magazine, vol. 12, no. 2, pp.** 34 **–51,**

1991. [27]

120**A. Turing,**"Computing machinery and intelligence," **Mind, vol. 59, no. 236, pp. 433–460, 1950.**

[28] P. Jackson, Introduction to expert system. Boston, MA, USA: Addison-Wesley, 1998. [29] J. H. Connell, Minimalist Mobile Robotics. Burlington, MA, USA: Morgan Kauf- mann, 1990. [30]

40**R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg,**"Dendral: A case study of the first expert system for scientific hypothesis formation," **Artificial Intelligence, vol. 61, no. 2, pp. 209 – 261,**

1993. [31] P. Salvaneschi, M. Cedei, and M. Lazzari, "Applying ai to structural safety moni- toring and evaluation," IEEE Expert, vol. 11, no. 4, pp. 24–34, Aug 1996.

91**[32] L. Zadeh,**"Fuzzy sets," **Information and Control, vol. 8, pp. 338–353, 1965. [33]** A. Cully, **J.** Clune, D. Tarapore, **and**

J.-B. Mouret, "Robots that can adapt like animals," Nature, vol. 521, no. 7553, pp. 503–507, 2015. [34] N. J. Nilsson, "Shakey the robot," SRI International Technical Note, Tech. Rep., 1984. [35] V. Dimitrov, M. DeDonato, A. Panzica, S. Zutshi, M. Wills, and T. Padir, "Hi- erarchical navigation architecture and robotic arm controller for a sample return rover," in Systems, Man, and Cybernetics (SMC), 2013 IEEE International Con- ference on, Oct 2013, pp. 4476–4481. [36]

63**R. Brooks,**"A robust layered control system for a mobile robot," **Robotics and Automation, IEEE Journal of, vol. 2, no. 1, pp. 14–23, Mar 1986.**

[37]

29**B. G. Buchanan and E. H. Shortliffe, Rule Based Expert Systems: The Mycin**

Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley Series in Artificial Intelligence). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984. [38]

206**R. A. Brooks,**"A robot that walks; emergent behaviors from a carefully evolved network," **Cambridge, MA, USA,**

Tech. Rep., 1989. [39]

205**M. Steinlechner** and **W. Parson,**"Automation and high through-put for a dna database laboratory: development of a laboratory information management sys- tem," **Croatian** medical journal, vol. **42,**

no. 3, pp. 252–255, 2001. [40]

17**A. Persidis,**"High-throughput screening," **Nature biotechnology, vol. 16, no. 5, pp. 488–493, 1998.**

[41]

2**K. E. Whelan and R. D. King,**"Intelligent software for laboratory automation,"

Trends in Biotechnology,

vol. 22, no. 9, pp. 440–445, 2004. [42]

3**N. Gauld and Gaston.,**"Driving miss daisy: The performance of an automated insect idenfitication system," **Hymenoptera: evolution, biodiversity and** biological-control, pp. **303–311, 2000.**

[43]

17**N. MacLeod, M. Benfield, and P. Culverhouse,**"Time to automate identification," **Nature, vol. 467, no. 7312, pp. 154–55, 2010.**

[44]

3**C. Darwin, On the Origin of Species.** England: **Dover Publications,**

1859. [45] D. M. M. and F. D. S., "Beneficial mutationselection balance and the effect of linkage on positive selection,"

207**Genetics, vol. 176,** no. 3, **pp. 1759–1798,**

2007. [46] L. S. Russell, "Body temperature of dinosaurs and its relationships to their ex- tinction," Journal of Paleontology, vol. 39, no. 3, pp. pp. 497–501, 1965. [47] B. Li and L. Tusheng, "Comments on coevolution in genetic algorithms," Computer Science, vol. 36, no. 4, pp. 34–63, 2009. [48]

101**J. H. Holland, Adaptation in Natural and Artificial Systems.** Boston, Mas-sachusetts: **MIT Press, 1992.** [49] M. **J.**

W. Lawrence J. Fogel, Alvin J. Owens,

195**Artificial Intelligence through Simu- lated Evolution.** Chichester, UK: **Wiley,**

1966. [50]

39**I. Rechenberg, Evolutionsstrategie: Optimierung** technischer **Systeme nach Prinzipien der** biologischen **Evolution.** Stuttgart: Fromman-Hozlboog **Verlag,** 1994. [51] **H.-P.Schwefel, Evolution and Optimum Seeking. New York,** NY, USA: **Wiley, 1995.**

[52]

149**J. Koza, Genetic Programming. Cambridge MA: MIT Press, 1992.**

[53] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," Nature, vol. 521, no. 7553, pp. 467–482, 2015. [54] C.

100**Rosin and R. Belew,**"New methods for competitive coevolution," **Evolutionary Computation, vol. 5, no.** 10, **pp. 1–29, 1997.**

[55]

68**J. Bongard and H. Lipson,**"Nonlinear system identification using coevolution of models and tests," **IEEE Trans. Evol. Comput., vol. 9, no. 4, pp. 361–384,**

2005. [56] Z. Hong and W. Jian, "A cooperative coevolutionary algorithm with application to job shop scheduling problem,"

75**in Service Operations and Logistics, and** Informat- ics, **2006. SOLI '06. IEEE International Conference on, June 2006, pp.**

746–751. [57]

34**K. C. Tan, Q. Yu, and J. H. Ang,**"A coevolutionary algorithm for rules discovery in data mining," **International Journal of Systems Science, vol. 37, no. 12, pp. 835–864, 2006.**

[58]

26**R. Dawkins and J. R. Krebs,**"Arms races between and within species," **Proceedings of the Royal Society of London. Series B. Biological Sciences, vol. 205, no. 1161, pp. 489–511, 1979.** [59] **J.** Cartlidge **and**

94**S. Bullock,**"Combating coevolutionary disengagement by reduc- ing parasite virulence," **Evolutionary Computation, vol. 12, no. 2, pp. 193–222, 2004.** [60] P. **J.** Angeline **and**

J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," Bibliometrics, vol. 155, no. 18, pp. 1–5, 1993. [61]

3**L. Panait and S. Luke,**"A comparative study of two competitive fitness functions," **in Proceedings of the Genetic and Evolutionary Computation Conference. Boston, Massachusetts: MIT Press,**

2002, pp. 567–573. [62]

3**T. Tan and J. Teo,**"Competitive coevolution with k-random opponents for pareto multiobjective optimization," **in Natural Computation, Third International** Con- ference **on,** 2007, pp. **63 – 67.**

[63]

86**D. B. Fogel,**"The advantages of evolutionary computation," **in Biocomputing and Emergent Computation: Proceedings of BCEC97. World Scientific Press, 1997,**

pp. 1–11. [64] H. GS, L. JD, and L. DS, "Computer-automated evolution of an x-band antenna for nasa's space technology 5 mission,"

171**Evolutionary Computation, vol.** 19, **no. 1, pp. 1–23,**

2011. [65]

108**R. Groß, K. Albrecht, W. Kantschik, and W. Banzhaf,**"Evolving chess playing programs," **in GECCO 2002,**

no. LSRO-CONF-2008-037. Morgan Kaufmann, 2002. [66]

33**O. E. David, H. J. van den Herik, M. Koppel, and N. S. Netanyahu,**

"Genetic algorithms for evolving computer chess programs,"

160**IEEE Transactions on** Evolu- tionary **Computation, vol.** 18, **no. 5, pp.**

779–789, 2014. [67]

45**H. Juille and J. B. Pollack, "Coevolving the**"ideal" **trainer: Application to the discovery of cellular automata rules," in** University **of**

Wisconsin. Morgan Kauf- mann, 1998, pp. 519–527. [68] C. Wang, S. Yu, W. Chen, and C. Sun, "Highly efficient light-trapping structure design inspired by natural evolution," Sci. Rep.,

192**vol. 3, no. 1, pp. 1–** 7, 2013. [69] **I.** Loshchilov **and**

T. Glasmachers. (2015) Black box optimization competition. [Online]. Available: http://bbcomp.ini.rub.de/ [70] M. Gauci, "Swarm robotic systems with minimal information processing," Ph.D. dissertation, The University of Sheffield, Sheffield, UK, 2014. [71]

5**R. Bellman, Dynamic** Progamming **and Lagrange Multipliers. Princeton, NJ, USA: Princeton University Press, 1957.**

[72]

56**N. Hansen, S. Muller, and P. Koumoutsakos,**"Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," **Evolutionary Computation, vol. 11, no. 1, pp. 1–18, March 2003.**

[73] J.

78**J. E. Dennis and J. J. Mor,**"Quasi-newton methods, motivation and theory," **SIAM Review,** vol. **19,** no. 1, **pp. 46–89, 1977.** [74] **J. R.**

Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Pittsburgh, PA, USA, Tech. Rep., 1994. [75]

57**C. M. Fonseca and P. J. Fleming,**"An overview of evolutionary algorithms in multiobjective optimization," **Evolutionary Computation, vol. 3, no. 1, pp. 1–16, 1995.** [76] S. Doncieux, **N.**

Bredeche, J.-B. Mouret, and A. G. Eiben, “Evolutionary robotics: What, why, and where to,” Frontiers in Robotics and AI, vol. 2, no. 4, 2015. [77] A.

204**Eiben,**“Grand challenges for evolutionary robotics,” **Frontiers in Robotics and AI,**

vol. 1, no. 4, 2014. [78]

3**D. Floreano and S. Nolfi,**“Adaptive behavior in competing co-evolving species,” **in The 4th European Conference on Artificial Life. MIT Press, 1997,**

pp. 378–387. [79]

71**D. Floreano, P.** Drr, **and C. Mattiussi,**“Neuroevolution: from architectures to learning,” **Evolutionary Intelligence, vol. 1, no. 1, pp. 47–62, 2008.**

[80]

76**K. O. Stanley and R. Miikkulainen,**“Evolving neural networks through augment- ing topologies,” **Evolutionary Computation, vol. 10, no. 2, pp. 99–127, 2002.**

[81]

41**A. L. Nelson, G. J. Barlow, and L. Doitsidis,**“Fitness functions in evolutionary robotics: A survey and analysis,” **Robotics and Autonomous Systems, vol. 57, no. 4, pp. 345 – 370,**

2009. [82]

38**D. Floreano and F. Mondada,**“Evolution of homing navigation in a real mobile robot,” **IEEE Trans.** Syst., **Man, and Cybernetics, Part B: Cybernetics, vol. 26, no. 3, pp. 396–407, 1996.**

[83] S.

25**Nolfi and D. Floreano, Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. MIT press,** 2000. [84] **B.** D.M. **and** O. **C.,**

“Understanding evolutionary potential in virtual cpu instruc- tion set architectures,” PLoS ONE, vol. 8, no. 12, p. e83242, 2013. [85] B. Batut, D. P. Parsons, S. Fischer, G. Beslon, and C. Knibbe, “In silico ex- perimental evolution: a tool to test evolutionary scenarios,” in Proceedings of the Eleventh Annual Research in Computational Molecular Biology (RECOMB) Satel- lite Workshop on Comparative Genomics. BioMed Central Ltd, 2013, pp. 1–6. [86] J.-M. Montanier and N. Bredeche, “Surviving the Tragedy of Commons: Emer- gence of Altruism in a Population of Evolving Autonomous Agents,” in

202**European Conference on Artificial Life,** Paris, France, Aug. **2011.**

[87] W. M, F.

196**D, and** K. **L,**"A quantitative test of hamilton's rule for the evolution of altruism," **PLoS Biology, vol. 9,**

no. 5, p. e1000615, 2011. [88]

49**D. Floreano, S. Mitri, S. Magnenat, and L. Keller,**"Evolutionary conditions for the emergence of communication in robots," **Current Biology, vol. 17, no. 6, pp. 514 – 519, 2007.**

[89] A. JE and B. JC, "Environmental influence on the evolution of morphological complexity in machines," PLoS Computational Biology, vol. 10, no. 1, p. e1003399, 2014. [90]

3**D. Cliff and G. F. Miller,**"Co-evolution of pursuit and evasion ii: Simulation meth- ods and results," **Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior,**

vol. 92, no. 2, pp. 101–106, 1995. [91]

3**D. Floreano,**"Evolutionary robotics in behavior engineering and artificial life," **in** Evolutionary **Robotics: From Intelligent Robots to Artificial Life. Applied AI Systems, 1998. Evolutionary Robotics Symposium. AAI Books, 1998.**

[92]

61**N. Jakobi, P. Husbands, and I. Harvey,**"Noise and the reality gap: the use of simulation in evolutionary robotics," **in Advances in Artificial Life: Proc. 3rd** European **Conf. Artificial Life.**

201**Springer-Verlag, 1995, pp. 704–720.**

[93] S.

21**Koos, J.-B. Mouret, and S. Doncieux,**"The transferability approach: Cross- ing the reality gap in evolutionary robotics," Evolutionary Computation, **IEEE** Transactions **on, vol. 17, no. 1, pp. 122–145,**

Feb 2013. [94] S.

203**Koos, A. Cully, and J. Mouret,**

"Fast damage recovery in robotics with the t-resilience algorithm," CoRR, vol. abs/1302.0386, 2013. [95] P. J. O'Dowd, M. Studley, and A. F. T. Winfield, "The distributed co-evolution of an on-board simulator and controller for swarm robot behaviours," Evol. Intell., vol. 7, no. 2, pp. 95–106, 2014. [96] G.

116**Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, "Au-**

tonomous evolution of gaits with the sony quadruped robot," in Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2. Citeseer, 1999, pp. 1297–1304. [97]

96V. Zykov, J. Bongard, and H. Lipson,"Evolving dynamic gaits on a physical robot," in Proc. 2004 Genetic and Evol. Computation Conf. ACM Press, Seattle, WA, 2004,

pp. 4722–4728. [98]

197R. Watson, S. Ficiei, and J. Pollack,

"Embodied evolution: embodying an evolu- tionary algorithm in a population of robots," in

19Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, vol. 1, 1999, pp.–342 Vol. 1.

[99] A.

54Eiben, E. Haasdijk, and N. Bredeche,"Embodied, On-line, On-board Evolu- tion for Autonomous Robotics," in Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution., ser. Series: Cognitive Systems Monographs,

S. K. E. P. Levi, Ed. Springer, 2010, vol. 7, pp. 361–382. [100] A.

104Eiben, S. Kernbach, and E. Haasdijk,"Embodied artificial evolution," Evolu- tionary Intelligence, vol. 5, no. 4, pp. 261–272, 2012.

[101] A. Eiben, N. Bredeche, M. Hoogendoorn, J.

127Stradner, J. Timmis, A. Tyrrell, A. Winfield, et al., "The triangle of life: Evolving robots in real-time and real- space," Advances in Artificial Life, ECAL,

pp. 1056–1063, 2013. [102] A. Eiben, "In vivo veritas: Towards the evolution of things," in Parallel Prob- lem Solving from Nature PPSN XIII, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, J. Branke, B. Filipi, and J. Smith, Eds. Springer Interna- tional Publishing, 2014, vol. 8672, pp. 24–39. [103] J. R. Tumbleston,

158D. Shirvanyants, N. Ermoshkin, R. Janusziewicz, A. R. Johnson,

D. Kelly, K. Chen, R. Pinschmidt, J. P. Rolland, A. Ermoshkin, E. T. Samulski, and J. M. DeSimone, "Continuous liquid interface production of 3d objects," Sci- ence, vol. 347, no. 6228, pp. 1349–1352, 2015. [104]

180L. Ljung,"System identification: Theory for the user," Englewood Cliffs, NJ: Prentice-Hall, 1999.

[105]

55J. Bongard and H. Lipson,"Nonlinear system identification using coevolution of models and tests," IEEE Transactions on Evolutionary Computation, vol. 9, no. 4, pp. 361–384, 2005.

[106]

44**D. B. Fogel, System identification through simulated evolution: a machine learning approach to modeling. Needham, MA,** USA: **Ginn Press, 1991.**

[107] D. Ljungquist and J. G. Balchen, “Recursive prediction error methods for on- line estimation in nonlinear state-space models,”

83**in Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on, Dec 1993, pp.** 714–719 **vol.**

1. [108]

19**E. J. Vladislavleva, G. F. Smits, and D. Den Hertog,**“Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming,” **Trans. Evol. Comp, vol. 13,** no. 2, **pp. 333–349,** Apr. **2009.** [109] A. Eiben **and**

112**J.** E. **Smith, Introduction to evolutionary computing.** Berlin, Hei- delberg: **Springer-** Verlag, 2003. [110] **J. Bongard and H. Lipson,**

“Automated damage diagnosis and recovery for remote robotics,” in Proc. 2004

36**IEEE Int. Conf.** Robot. **and** Autom. **IEEE Computer Society Press,**

New Orleans, LA, 2004, pp. 3545–3550. [111] ——,

52**“Automated robot function recovery after unanticipated failure or environ-**

mental change using a minimum of hardware trials,” in Proc. 2004 NASA/DoD Conf. Evolvable Hardware. IEEE Computer Society Press, Los Alamitos, CA, 2004, pp. 169–176. [112] S.

1**Koos, J. Mouret, and S. Doncieux,**“Automatic system identification based on coevolution of models and tests,” **in** Proc. **2009 IEEE** Congr. Evol. **Computation.**

IEEE Press, Trondheim, Norway, 2009, pp. 560–567. [113] M. Mirmomeni and W. Punch, “Co-evolving data driven models and test data sets with the application to forecast chaotic time series,” in

10**Proc. 2011 IEEE Congr. Evol. Comput. IEEE** Press, **New Orleans,**

LA, USA, 2011, pp. 14–20. [114] D. Le Ly and H. Lipson, “Optimal experiment design for coevolutionary active learning,”

21**IEEE Trans. Evol.** Computation, **vol.** 18, **no. 3, pp.**

394–404, 2014. [115]

2**B. Kouchmeshky, W. Aquino, J. C. Bongard, and H. Lipson, “Co-evolutionary**

algorithm for structural damage identification using minimal physical testing," International Journal for Numerical Methods in Engineering, vol. 69, no. 5, pp. 1085–1107, 2007. [116]

2**M. Mirmomeni and W. Punch, "Co-evolving data driven models and test data sets**

with the application to forecast chaotic time series," in 2011 IEEE Congress on Evolutionary Computation. Auburn University, New Orleans, LA, 2011, pp. 14 –20. [117]

62**J. Bongard, V. Zykov, and H. Lipson,**"Resilient machines through continuous self-modeling," Sci., **vol. 314, no. 5802, pp. 1118–1121, 2006.** [118] S. Koos, **J.** B. Mouret, **and**

21**S. Doncieux,**"The transferability approach: Crossing the reality gap in evolutionary robotics," **IEEE Trans. Evol.** Computation, **vol. 17, no. 1, pp. 122–145,**

Feb 2013. [119] P. J. O'Dowd, M. Studley, and A. F. T. Winfield, "The distributed co-evolution of an on-board simulator and controller for swarm robot behaviours," Evol. Intell., vol. 7, no. 2, pp. 95–106, 2014. [120] B. Hedwig and

210**J. F. A.** Poulet, "Complex auditory behaviour emerges from simple reactive steering," **Nature,**

vol. 430, no. 7001, pp. 781–785, 2004. [121]

3**E. Baird, M. J. Byrne, J. Smolka, E. J. Warrant, and M. Dacke,**"The dung beetle dance: An orientation behaviour?" **PLoS ONE,**

vol. 7, no. 1, p. e30211, 01 2012. [122]

3**M. D. M. Byrne,**"Visual cues used by ball-rolling dung beetles for orientation," **Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and** Be- havioral **Physiology,**

vol. 189, no. 6, pp. 411–418, 2003. [123]

3**E. G. Matthews,**"Observations on the ball-rolling behavior of canthon pilularius," **Psyche,** pp. **75–93, 1963.**

[124]

111**I. Rano,**"A steering taxis model and the qualitative analysis of its trajectories," **Adaptive Behavior, vol. 17, no. 3, pp. 197–211, 2009.**

[125] S.

24**Garnier, J. Gautrais, and G. Theraulaz,**"The biological principles of swarm intelligence," **Swarm Intelligence, vol. 1, no. 1, pp. 3–31, 2007.** [126] **C. W.**

Reynolds,

“Flocks, herds, and schools: A distributed behavioral model,”

132**Computer Graphics, vol. 21, no. 4, pp. 25–34, 1987.**

[127]

12**R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz,**“Self-organized aggregation in cockroaches,” **Animal Behaviour, vol. 69, no. 1, pp. 169 – 180,**

2005. [128]

82**C. R. Carroll** and **D. H. Janzen,**“Ecology of foraging by ants,” Annu. **Review of Ecology and Systematics, vol. 4, pp. 231–257,**

1973. [129]

5**J. E. Lloyd,**“Bioluminescent communication in insects,” **Annual Review of Entomology, vol. 16, pp. 97–122, 1971.**

[130]

5**O. H. Bruinsma,**“An analysis of building behaviour of the termite macrotermes subhyalinus (rambur),” **Ph.D.** dissertation, **Wageningen** University, **Wageningen, The Netherlands, 1979.**

[131]

103**M. Dorigo and L. Gambardella,**“Ant colony system: a cooperative learning approach to the traveling salesman problem,” Evolutionary Computation, **IEEE Transactions on, vol. 1,** no. 1, **pp. 53–66,**

1997. [132]

28**J. Kennedy and R. Eberhart,**“Particle swarm optimization,” **in Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4,** Nov 1995, **pp. 1942– 1948 vol.4.** [133] **M. Dorigo, M. Birattari,**

139**C. Blum, M. Clerc, T. Stützle, and A. Winfield,**

Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings. Springer, 2008, vol. 5217. [134]

14**O. Holland and C. Melhuish,**“Stigmergy, self-organization, and sorting in collective robotics,” **Artificial Life, vol. 5, no. 2, pp. 173–202, 1999.**

[135]

81**G. Di Caro and M. Dorigo,**"Antnet: Distributed stigmergetic control for communications networks," **J. Artif.** Int. **Res., vol. 9,** no. 1, **pp. 317–365, Dec. 1998.**

[136]

208**K. Socha,**"Aco for continuous and mixed-variable optimization," **in Ant Colony**

Optimization and Swarm Intelligence, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Sttzle, Eds. Springer Berlin Heidelberg, 2004, vol. 3172, pp. 25–36. [137] J.

198**Bjerknes and A. T. Winfield,**"On fault tolerance and scalability of swarm robotic systems," **in**

Distributed Autonomous

93**Robotic Systems, ser. Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, 2013, vol. 83,**

pp. 431–444. [138]

18**J. Chen, M. Gauci, W. Li, A. Kolling, and R. Gros,**"Occlusion-based cooperative transport with a swarm of miniature mobile robots," **Robotics, IEEE Transactions on, vol. 31, no. 2, pp. 307–321,**

April 2015. [139]

5**M. Gauci, J. Chen, T. Dodd, and R. Groß, "Evolving aggregation behaviors in**

multi-robot systems with binary sensors," in Distributed Autonomous Robotic Sys- tems, ser. Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, 2014, vol. 104, pp. 355–367. [140] E. ahin, "Swarm robotics: From sources of inspiration to domains of application,"

48**in Swarm Robotics,** ser. **Lecture Notes in Computer Science, E.** ahin **and W. Spears, Eds.** Springer **Berlin Heidelberg,** 2005, vol. 3342, **pp. 10–20.** [141] C. Blum **and**

R. Groß, "Swarm intelligence in optimization and robotics," in Hand- book of Computational Intelligence,

10**J. Kacprzyk** and **W. Pedrycz,** Eds. **Springer, Berlin, Heidelberg,** 2015, **pp.**

1293–1311. [142]

42**B. Gerkey and M. Mataric,**"Sold!: auction methods for multirobot coordination," **Robotics and Automation, IEEE Transactions on, vol. 18, no. 5, pp. 758–768, Oct 2002.**

[143]

99A. F. T. Winfield, “Distributed sensing and data collection via broken ad hoc

wireless connected networks of mobile robots,” in Distributed Autonomous Robotic Systems 4, L. Parker, G. Bekey, and J. Barhen, Eds. Springer Japan, 2000, pp. 273–282. [144]

15V. Trianni, R. Gro, T. Labella, E. ahin, and M. Dorigo, “Evolving aggregation

behaviors in a swarm of robots,” in Advances in Artificial Life, ser. Lecture Notes in Computer Science, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. Kim, Eds. Springer Berlin Heidelberg, 2003, vol. 2801, pp. 865–874. [145] S.

12Garnier, C. Jost, J. Gautrais, M. Asadpour, G. Caprari, R. Jeanson, A. Grimal, and G. Theraulaz,“The embodiment of cockroach aggregation behavior in a group of micro-robots,” Artificial Life, vol. 14, no. 4, pp. 387–408,

Oct. 2008. [146]

95A. Howard, M. J. Matarić, and G. S. Sukhatme, “Mobile sensor network deploy-

ment using potential fields: A distributed, scalable solution to the area coverage problem,” in Distributed Autonomous Robotic Systems 5. Springer, 2002, pp. 299–308. [147]

70J. McLurkin and J. Smith,“Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots,” in in 7th International Symposium on Distributed Autonomous Robotic Systems (DARS. Citeseer, 2004.

[148]

184K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura,

“Self-organizing for- mation algorithm for active elements,” in Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on, 2002, pp. 416–421. [149]

5J. Chen, M. Gauci, M. J. Price, and R. Groß, “Segregation in swarms of e-puck

robots based on the brazil nut effect,” in Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, ser. AA- MAS ’12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 163–170. [150] A.

165Turgut, H. elikkanat, F. Gke, and E. ahin,

“Self-organized flocking in mobile robot swarms,”

122Swarm Intelligence, vol. 2, no. 2-4, pp. 97–120, 2008. [151] E.

B. C.R. Kube, “Collective robotics: from social insects to robots,”

14**Adaptive Behavior, vol. 2, no. 2, pp. 189–** 218, **1993.** [152] **C. Kube and E. Bonabeau,**“Cooperative transport by ants and robots,” **Robotics and Autonomous Systems, vol. 30, no.** 12, **pp. 85 – 101, 2000.** [153] **R.** Gross **and**

M. Dorigo, “Towards group transport by swarms of robots,”

126**Int. J. Bio-Inspired** Comput., **vol. 1,** no. **1/2, pp. 1–13,**

Jan. 2009. [154]

87**J. Werfel, K. Petersen,** and **R. Nagpal,**“Designing collective behavior in a termite-inspired robot construction team,” **Science, vol. 343, no. 6172, pp. 754–758, 2014.**

[155] S. Camazine,

141**Self-organization in biological systems. Princeton University Press, 2003.**

[156]

115**B. Webb,**“What does robotics offer animal behaviour?” **Animal Behaviour, vol. 60, no. 5, pp. 545 – 558, 2000.**

[157] ——, “Using robots to model animals: a cricket test,”

130**Robotics and Autonomous Systems, vol. 16, no.** 2134, **pp. 117 – 134,**

1995. [158] A. Popov and V. Shuvalov, “Phonotactic behavior of crickets,” J. of Comparative Physiology, vol. 119, no. 1, pp. 111–126, 1977. [159]

3**A. M. Farah and T. Duckett,**“Reactive localisation of an odour source by a learning mobile robot,” **in In Proceedings of the Second Swedish Workshop on Autonomous Robotics. SWAR Stockholm, Sweden, 2002,**

pp. 29–38. [160] A.

3**Lilienthal and T. Duckett,**“Experimental analysis of smelling braitenberg vehi- cles,” **in In Proceedings of the ieee international conference on advanced robotics. Coimbra, Portugal, 2003,**

pp. 58–63. [161]

92**T. Balch, F. Dellaert, A. Feldman, A. Guillory, C. Isbell, Z. Khan, S. Pratt,**

A. Stein, and H. Wilde, “How multirobot systems research will accelerate our understanding of social animal behavior,” Proceedings of the IEEE, vol. 94, no. 7, pp. 1445 –1463, 2006. [162]

3J. Chappell and S. Thorpe,"Ai-inspired biology: Does ai have something to contribute to biology?" Proceedings of the International Symposium on AI Inspired Biology: A Symposium at the AISB 2010 Convention, Leicester, UK, 2010.

[163]

200J. Faria, J. Dyer, R. Clément,

et al., "A novel method for investigating the collec- tive behaviour of fish: Introducing 'robofish'," 144Behavioral Ecology and Sociobiol- ogy, vol. 64, no. 8, pp. 1211–1218, 2010.

[164]

1J. Halloy, F. Mondada, S. Kernbach, et al., "Towards bio-hybrid systems made

of social animals and robots," in Biomimetic and Biohybrid Systems, ser. Lecture Notes in Comput. Sci. Springer, Berlin, Heidelberg, Germany, 2013, vol. 8064, pp. 384–386. [165] J. Halloy, G. Sempo1, G. Caprari,

23et al.,"Social integration of robots into groups of cockroaches to control self-organized choices," Sci., vol. 318, no. 5853, pp. 1155– 1158, 2007.

[166] T. Schmickl, S. Bogdan, L. Correia,

1et al.,"Assisi: Mixing animals with robots in a hybrid society," in Biomimetic and Biohybrid

Systems,

124ser. Lecture Notes in Comput. Sci. Springer, Berlin, Heidelberg, Germany, 2013, vol. 8064, pp.

441–443. [167]

7R. Vaughan, N. Sumpter, J. Henderson,

et al., "Experiments in automatic flock control," Robot.

138and Autonomous Syst., vol. 31, no. 1, pp. 109–117, 2000.

[168]

23J. Krause, A. F. Winfield, and J.-L. Deneubourg,"Interactive robots in experimental biology," Trends in Ecology and Evolution, vol. 26, no. 7, pp. 369 –375, 2011.

[169] S. G. Halloy J., “Social integration of robots into groups of cockroaches to control self-organized choices,”

32**Science, vol. 318, no. 5853, pp. 1155–1158, 2007.** [170] **J. Krause, A.** F. **Winfield, and J.-L. Deneubourg,**“Interactive robots in experi- mental biology,” **Trends in Ecology and Evolution,** vol. **26,** no. **7,**

pp. 369 – 375, 2011. [171] V. Kopman, J. Laut, G. Polverino, et al., “Closed-loop control of zebrafish response using a bioinspired robotic-fish in a preference test,” J. of The Roy. Soc. Interface, vol. 10, no. 78, pp. 1–8, 2013. [172]

3**R. Vaughan, N. Sumpter, A. Frost, and S. Cameron,**“Robot sheepdog project achieves automatic flock control,” **The fourth international conference on** Au- tonomous **agents,** pp. **489–493, 1998.**

[173]

1**A. Gribovskiy, J. Halloy, J.-L. Deneubourg, H. Bleuler, and F. Mondada, “Towards**

mixed societies of chickens and robots,” in Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. Boston, Massachusetts: MIT press, 2010, pp. 4722 –4728. [174] A.

121**M. Turing,**“Computing machinery and intelligence,” **Mind, vol. 59, no. 236, pp. 433–460, 1950.**

[175] S. Harnad, “Minds, machines and turing: The indistinguishability of indistinguish- ables,” J.

119**Logic, Language and** Inform., **vol. 9, no. 4, pp. 425–445, 2000.**

[176]

123**J. L. Elman,**“Finding structure in time,” **Cognitive Sci., vol. 14,** no. 2, **pp. 179–211, 1990.**

[177]

60**H.-G. Beyer and H.-P. Schwefel,**“Evolution strategies - a comprehensive introduc- tion,” **Natural Computing, vol. 1, no. 1, pp. 3–52, 2002.**

[178]

43**X. Yao, Y. Liu, and G. Lin,**“Evolutionary programming made faster,” **IEEE Trans.** on **Evol. Comput., vol. 3, no. 2, pp. 82–102, 1999.**

[179]

106**F. Mondada, M. Bonani, X. Raemy,** et al., **“The e-puck, a robot designed for**

education in engineering,” in Proc. 9th Conf. on Autonomous Robot Systems and Competitions, vol. 1.

IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65. [180] S.

24**Magnenat, M. Waibel, and A. Beyeler,**"Enki: The fast 2D robot simulator," **http://home.gna.org/enki/,**

2011. [181]

37**R. L. Graham and N. J. A. Sloane,**"Penny-packing and two-dimensional codes," **Discrete and Computational Geometry, vol. 5, no. 1, pp. 1–11, 1990.** [182] P. Levi **and**

128**S. Kernbach, Symbiotic Multi-Robot Organisms: Reliability,** Adapt- ability, **Evolution.** Berlin, Heidelberg: **Springer-Verlag,**

2010. [183] B. Eldridge and A. Maciejewski, "Limited bandwidth recognition of collective behaviors in bio-inspired swarms," in Proc. 2014 Int. Conf.

199**Autonomous Agents and Multi-Agent** Syst. IFAAMAS **Press,**

Paris, France, 5 2014, pp. 405–412. [184]

53**G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, CA: O'Reilly Media,**

2008. [185]

27**M.-K. Hu,**"Visual pattern recognition by moment invariants," **IRE Transactions on Information Theory, vol. 8, no. 2, pp. 179–187, 1962.** [186] W. Li, **M.** Gauci, J. Chen, **and**

22**R. Groß,**"Online supplementary material," **http: //naturalrobotics.group.shef.ac.uk/supp/**

2015-002/, 2014. [187]

13**R. D. King, J. Rowland, et al.,**"The automation of science," Sci., **vol. 324, no. 5923, pp. 85–89, 2009.** [188] **M. Schmidt and H. Lipson,**"Distilling free-form natural laws from experimental data," Sci., **vol. 324, no. 5923, pp. 81–85, 2009.**

[189]

72**J. Bongard and H. Lipson,**"Active coevolutionary learning of deterministic finite automata," The **Journal of Machine Learning Research, vol. 6, pp. 1651–1678, 2005.**

[190]

3**E. Martin, Macmillan Dictionary of Life Sciences (2nd ed.).** London: **Macmillan**

Press,

1983. [191]

3**M. Dacke, M. J. Byrne, C. H. Scholtz, and E. J. Warrant,**"Lunar orientation in a beetle," Proc. **of the** Roy. Soc. **of London. Series B: Biological** Sci., vol. **271,**

no. 1537, pp. 361–365, 2004. [192]

3**E. Baird, M. J. Byrne, J. Smolka, E. J. Warrant, and M. Dacke,**"The dung beetle dance: an orientation behaviour?" **PLoS ONE,**

vol. 7, no. 1, p. e30211, 2012. [193]

2**L. Grossman,**"Computer literacy tests: Are you human?" **June 2008.**

[Online]. Available:

162**http://www.time.com/time/** magazine/article **/0,** 9171,1812084 **,00.html**

8 12 14 16 18 20 22 24 26 28 30 34 36 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 76 78 80 82 84 86 88 90 92 94 96 98 100 102 104 106 108 110 112 114 116 118 120 122 124 126 128 132 136 138 140 142 144 146 148 150 152 1

84**Introduction 1 Introduction 1 Introduction 2 Background and Related Work 2 Background and Related Work 2.**

2 Evolutionary Computation

5**2 Background and Related Work 2.** 2 **Evolutionary Computation** 2 Background **and** Related Work **2.**

2 Evolutionary Computation

5**2 Background and Related Work 2.** 2 **Evolutionary Computation** 2 Background **and** Related Work **2.**

2 Evolutionary Computation

5**2 Background and Related Work 2.** 2 **Evolutionary Computation**

5**2 Background and Related Work 2.** 2 **Evolutionary Computation** 2 Background **and** Related Work **2 Background and Related Work 2.3** Combining AI/Robotics **and** Animal Behavior **2 Background and Related Work 2.3** Combining AI/Robotics **and** Animal Behavior **2 Background and Related Work 2.**

3 Combining AI/Robotics and Animal Behavior 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.1 Methodology 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.1 Methodology 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.1 Methodology 3

Reverse Engineering Swarm Behaviors Through Turing Learning 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3.3 Simulation Results 3 Reverse Engineering Swarm Behaviors Through Turing Learning 3 Reverse Engineering Swarm Behaviors Through Turing Learning 4 A Real-World Validation of Turing Learning 4 A Real-World Validation of Turing Learning 4 A Real-World Validation of Turing Learning 4 A Real-World Validation of Turing Learning 4 A Real-World Validation of Turing Learning 4.5 Results 4 A Real-World Validation of Turing Learning 4.5 Results 4 A Real-World Validation of Turing Learning 4.5 Results 4 A Real-World Validation of Turing Learning 4.5 Results 4 A Real-World Validation of Turing Learning 4 A Real-World Validation of Turing Learning 5 Inferring Individual Behaviors Through Interactive Turing Learning 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.3 Case Study One 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.4 Case Study Two 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.4 Case Study Two 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.4 Case Study Two 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.4 Case Study Two 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.4 Case Study Two 5 Inferring Individual Behaviors Through Interactive Turing Learning 5.4 Case Study Two 5 Inferring Individual Behaviors Through Interactive Turing Learning

77**Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography Bibliography**

3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 91 93 95 97 99 101 103 105 107 109 111 113 115 117 121 123 125 127 129 131 133 135 137 139 141 143 145 147 149 151 153