Preference-Based Many-Objective Evolutionary Testing Generates Harder Test Cases for Autonomous Agents

Sabrine Kalboussi[1], Slim Bechikh[1], Marouane Kessentini[2], and Lamjed Ben Said[1]

[1] SOIE Lab, ISG-Tunis, University of Tunis, Tunisia
{sabrine.kalboussi,slim.bechikh}@gmail.com,
lamjed.bensaid@isg.rnu.tn
[2] CS, Missouri University of Science and Technology, Missouri, USA
marouanek@mst.edu

Abstract. Despite the high number of existing works in software testing within the SBSE community, there are very few ones that address the problematic of agent testing. The most prominent work in this direction is by Nguyen et al. [13], which formulates this problem as a bi-objective optimization problem to search for hard test cases from a robustness viewpoint. In this paper, we extend this work by: (1) proposing a new seven-objective formulation of this problem and (2) solving it by means of a preference-based many-objective evolutionary method. The obtained results show that our approach generates harder test cases than Nguyen et al. method ones. Moreover, Nguyen et al. method becomes a special case of our method since the user can incorporate his/her preferences within the search process by emphasizing some testing aspects over others.

Keywords: Agent testing, many-objective optimization, user's preferences.

1 Introduction

Software testing is a software development phase that aims to evaluate the product quality and enhance it by detecting errors and problems [11]. Despite the big efforts performed in the software testing research field, such activity remains complex and so expensive from the effort viewpoint and also the cost one. For this reason, researchers have proposed a new testing approach called *Search-Based Software Testing* (*SBST*) [7]. The latter consists in modeling the software testing problem as an optimization problem and then solving it by using a particular search method (usually metaheuristic) such as Evolutionary Algorithms (EAs), tabu search, etc. The SBST approach is a sub area of Search Based Software Engineering (SBSE) [7] and it is applied to several testing types [9], [10]. It has shown a great effectiveness and efficiency in achieving the testing goals.

There is little work in testing autonomous agents in regard what has already achieved in software testing [4], [14]. The pro-activity characteristic of software agent makes its test a hard task since it may react in different manners for the same input over time. Recently, Nguyen et al. [13] have proposed a search-based method to test autonomous agents. In fact, Nguyen et al. identified two soft goals that are *robustness* and *efficiency* to test an autonomous Cleaner agent that has to keep clean a square area of an airport. However, they detailed/used only the robustness soft goal by proposing only two objective functions that drive the used multi-objective EA (i.e., NSGA-II [5]). The proposed Evolutionary Testing (ET) method has demonstrated its effectiveness in finding test cases that reply to the two considered objective functions belonging to the robustness soft goal.

In this paper, we propose a Preference-based Many-Objective Evolutionary Testing (P-MOET) method which corresponds to an extension of Nguyen et al.'s work [13]. The main idea is to propose additional objective functions corresponding to different soft goals with the aim to generate *harder* test cases (a hard test case is a test case that urges the agent to not achieve the soft goals) than Nguyen et al.'s method ones, and then solving the new obtained problem by considering all objective functions *simultaneously*. The main contributions of this work are as follows: (1) proposing five additional objective functions corresponding to different soft goals, (2) solving the obtained problem in a many-objective fashion [8], (3) Offering the user the ability to incorporate his/her preferences by emphasizing some objectives (testing aspects) over others by means of the use of the reference point concept [3], [2] and (4) outperforming in average an existing approach [13] when generating test cases on the same experimental environment.

2 Proposed Approach

2.1 New Many-Objective Formulation for Autonomous Agent Testing Problem

We propose a new many-objective formulation for the ET problem that considers simultaneously *seven* objectives. This formulation is stated as follows:

$$Min\, f(u) = [f_1(u), f_2(u), ..., f_7(u)]^T \tag{1}$$

where the test case u is encoded as follows $u = (x_1, y_1, x_2, y_2, ..., x_n, y_n)$ such that x_i and y_i are respectively the abscissa and the ordinate of the object i in the area to clean. Table 1 summarized the seven used objective functions (the two first ones are defined in [13] and the five new ones are proposed by us). The latter are classified into four families based on the ensured soft goal where *TPC*(u) is the Total Power Consumption, *NEO*(u) is the Number of Encountered Obstacles, *ACW*(u) is the Amount of Collected Waste during a specified amount of time that we fixed to 10 seconds in this work, *NDNW*(u) is the Number of times that the agent Drops to collect the Nearest Waste, *NDND*(u) is the Number of times that the agent Drops to put the waste in the Nearest Dustbin, *NDNC*(u) is the Number of times that the agent Drops to go to the Nearest Charging station when its battery is low, and *NRSD*(u) is the Number of times that the agent does not Respect the Safety Distance. All these quantities are computed when executing the Cleaner agent on the test case u. The expected outcomes are hard test cases that obstruct the agents under test from reaching the soft-goals under consideration. When multiple soft-goals are considered at once, we expect to obtain test cases that satisfy multiple *hard-to-find conditions* simultaneously.

Table 1. Classification of used objectives with their objective functions

Soft goal family	*Objective*	*Objective function*
Robustness	Maintain battery	$f_1(u) = 1/TPC(u)$
	Avoid obstacle	$f_2(u) = 1/NEO(u)$
Efficiency	Clean up waste	$f_3(u) = ACW(u)$
Stability	Collect the nearest waste	$f_4(u) = 1/NDNW(u)$
	Put the waste into the nearest dustbin	$f_5(u) = 1/NDND(u)$
	Go to the nearest charging station	$f_6(u) = 1/NDNC(u)$
Safety	Maintain a separation distance from obstacle	$f_7(u) = 1/NRSD(u)$

2.2 P-MOET: Preference-Based Many-Objective Evolutionary Testing Method

To solve our many-objective ET problem, we use a proposed preference-based MOEA, called *r-NSGA-II* (reference solution-based NSGA-II) [3]. Fig. 1 describes the adaptation of r-NSGA-II algorithm to our many-objective autonomous agent testing problem. The algorithm basic iteration begins by generating the parent population P_t of size N. After that, each test case from P_t is executed on the Cleaner agent module. Once all P_t test cases are evaluated, we apply binary tournament selection and genetic operators (crossover and mutation) to generate the offspring population Q_t of

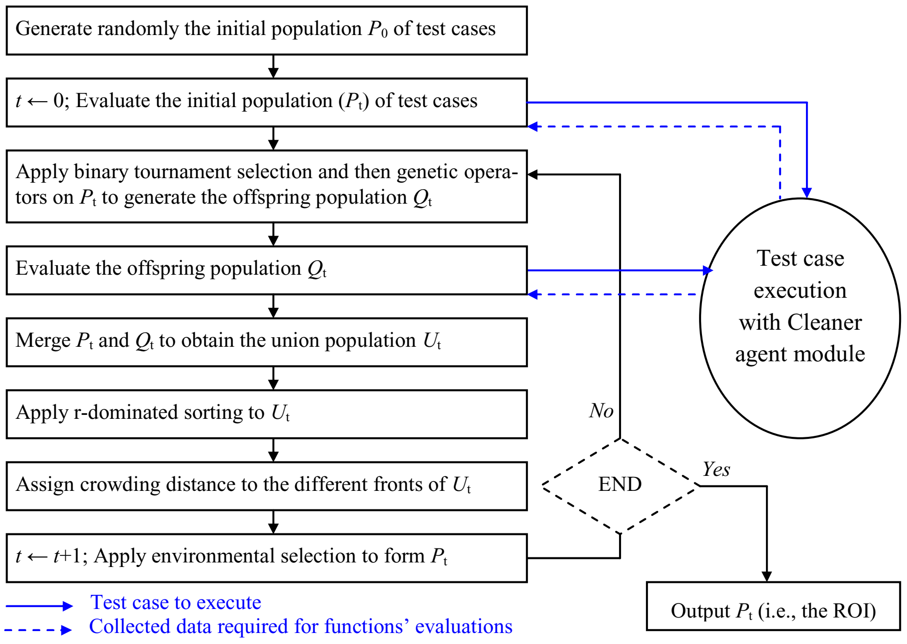
Generate randomly the initial population P_0 of test cases
$t \leftarrow 0$; Evaluate the initial population (P_t) of test cases
Apply binary tournament selection and then genetic operators on P_t to generate the offspring population Q_t
Evaluate the offspring population Q_t
Merge P_t and Q_t to obtain the union population U_t
Apply r-dominated sorting to U_t
Assign crowding distance to the different fronts of U_t
$t \leftarrow t+1$; Apply environmental selection to form P_t
Test case execution with Cleaner agent module
No
END
Yes
Output P_t (i.e., the ROI)
Test case to execute
Collected data required for functions' evaluations

Fig. 1. Adaptation schema of r-NSGA-II to the many-objective agent testing problem

size N. We form U_t by merging P_t and Q_t (the size of U_t is then $2N$). We apply non-r-dominated sorting based on the r-dominance relation to sort U_t into several non-r-dominated fronts. After that, P_{t+1} is filled with individuals of the best non-r-dominated fronts, one at a time. Since overall population size is $2N$, not all fronts may be accommodated in N slots available in the new population P_{t+1}. When the last allowed front is being considered, it may contain more solutions than the remaining available slots in P_{t+1}. Hence, only least crowded solutions of the last considered front are saved and the others are rejected. Concerning the Cleaner agent module, we have downloaded it from its authors' Web site via the URL [12], and then we have *adjusted* it to our work. We note that the overall adaptation of r-NSGA-II to the many-objective autonomous agent testing problem is named *P-MOET*.

3 Experimental Study

When generating test cases for agents based on the considered soft goals or goals (objectives), the user may prefer: (1) emphasizing some soft goals/goals over others or (2) emphasizing all soft goals/goals simultaneously. For this reason, different scenarios are used and summarized in table 2.

Table 2. Used scenarios with their reference points

Scenario N°	*Preferred soft goal(s)*	*Used reference point*
1	Robustness	A = (**0.1**, **0.1**, 0.9, 0.7, 0.8, 0.7, 0.8)
2	Efficiency	B = (0.7, 0.8, **0.1**, 0.6, 0.8, 0.7, 0.8)
3	Stability	C = (0.7, 0.8, 0.9, **0.1**, **0.1**, **0.1**, 0.8)
4	Safety	D = (0.7, 0.8, 0.9, 0.7, 0.8, 0.6, **0.1**)
5	All soft goals	E = (**0.1**, **0.2**, **0.1**, **0.2**, **0.2**, **0.1**, **0.1**)

Table 3. Median values of the test case hardness metrics for the 5 P-MOET scenarios and Nguyen et al. work [13] over 51 runs on Cleaner agent module. The *p-values* of the Wilcoxon test, with $\alpha = 0.05$, have shown that all the results are statistically different from each others. Best values regarding hardness are mentioned in bold.

Scenario N°	*TPC*	*NEO*	*ACW*	*NDNW*	*NDND*	*NDNC*	*NRSD*
1	**335**	**4**	2	1	1	1	4
2	300	2	**1**	0	0	1	0
3	295	1	2	**4**	**3**	**3**	2
4	250	3	2	2	1	1	**8**
5	330	3	2	3	2	2	6
Nguyen et al.	**335**	**4**	2	1	0	0	4

For instance, for *scenario 1*, we emphasize only the objectives related to the *robustness* soft goal with the aim to generate test cases that consume so much power and increase the probability of crashing with the different existing obstacles. This is achieved by using the reference point A= (**0.1**, **0.1**, 0.9, 0.7, 0.8, 0.7, 0.8) where we emphasize just the *two first* objectives (mentioned with bold numbers) by fixing them

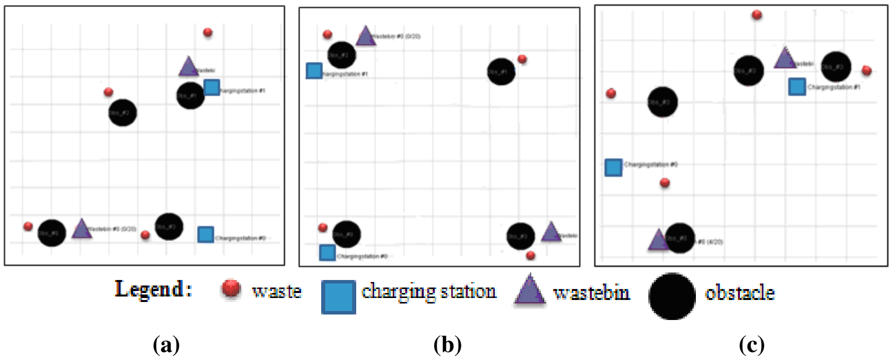

Fig. 2. Representative test cases for: (a) scenario 3, (a) scenario 5 and (b) Nguyen et al. work

to a low value (< 0.4) and the other aspiration (i.e., desired) values are set to high values (> 0.6). We restricted each objective value to the normalized interval [0,1] in order to ease the expression of user's preferences. For each scenario, we run the P-MOET method with a population of 50 individuals for 100 generations. Then we select *randomly* a representative test case from the obtained ROI [1]. The difficulties related to the obtained test cases are assessed based on the following metrics: (1) *TPC*, (2) *NEO*, (3) *ACW*, (4) *NDNW*, (5) *NDND*, (6) *NDNC* and (7) *NRSD*. In fact, each scenario test case is executed 51 times on the Cleaner agent module and the different metrics' values are recorded. After that, we record the median values in table 3. Due to the stochastic aspects of the results, we use the Wilcoxon rank sum hypothesis test in a pairwise comparison fashion, with a 95% confidence level ($\alpha = 0.05$), in order to detect whether the results are significant or not. According to table 3, we observe that the P-MOET method has demonstrated its ability and flexibility in replying to the requirements of each scenario. For example, scenario 3 has the maximal values regarding the stability metrics (*NDNW*, *NDND* and *NDNC*). This observation shows the effectiveness of our method in solving the 7-objective agent testing problem while respecting user's preferences. Similar observations are obtained for the scenarios 1, 2 and 4. Although scenario 5 does not present any best value regarding the test case hardness metrics, it seems to be the scenario that provides the hardest test case set. In fact, this scenario emphasizes all the soft goals simultaneously, which is demanding. Consequently, it makes a compromise between: (1) robustness, (2) efficiency, (3) stability and (4) safety. For this reason, it furnishes the hardest test cases. This result is emphasized by fig. 2(a-b) which illustrates two test cases for the third scenario and the fifth one.

Next, we compare the P-MOET scenario 5 against Nguyen et al.'s work. From table 3, we remark that Nguyen et al. method's representative test case has the maximal values for the *TPC* and *NEO* metrics. The same observation is obtained for scenario 1. We can say that the P-MOET can reproduce the Nguyen et al. method test cases by running scenario 1. Fig. 2(c) illustrates the test case of Nguyen et al. method. When comparing the latter with the test case of P-MOET scenario 5, we remark that although the fifth scenario test case does not present any best value in terms of the considered hardness metrics in table 3, we observe from fig. 2(b-c) that the scenario 5 test case is harder than Nguyen et al. method one. This is illustrated by the locations of the different objects in the environment for each test case in regard to the semantic meaning of objects' locations.

4 Conclusions and Future Works

In this paper, we have proposed, for the first time, a preference-based many-objective method for autonomous agent testing. The latter has shown its ability to generate harder test cases than Nguyen et al. method ones. The obtained results confirm the effectiveness of our method. For future research, we plan to extend this work to test a multi-agent system in an attempt to evaluate cooperation, coordination and competition between different agents that can have different internal architectures. Moreover, it would be interesting to analyze test cases residing in knee regions [2] where the trade-offs between the testing objectives are maximal.

References

1. Adra, S.F., Griffin, I., Fleming, P.J.: A Comparative Study of Progressive Preference Articulation Techniques for Multiobjective Optimisation. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 908–921. Springer, Heidelberg (2007)
2. Bechikh, S., Ben Said, L., Ghédira, K.: Searching for Knee Regions of the Pareto Front using Mobile Reference Points. Soft Computing 15(9), 1807–1823 (2011)
3. Ben Said, L., Bechikh, S., Ghédira, K.: The r-Dominance: A New Dominance Relation for Interactive Evolutionary Multicriteria Decision Making. IEEE Trans. on Evolutionary Computation 14(5), 801–818 (2010)
4. Coelho, R., Kulesza, U., Staa, A., Lucena, C.: Unit Testing in Multi-agent Systems using Mock Agents and Aspects. In: International Workshop on Software Engineering for Large-Scale Multi-agent Systems, pp. 83–90 (2006)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6(2), 182–197 (2002)
6. Harman, M., Ph, U., Jones, B.F.: Search-Based Software Engineering. Information and Software Technology 43, 833–839 (2001)
7. Harman, M., Mansouri, S.A., Zhang, Y.: Search-Based Software Engineering: Trends, Techniques and Applications. ACM Computing Surveys 45(1), 11 (2012)
8. Hughes, E.J.: Evolutionary Many-objective Optimization: Many Once or One Many? In: IEEE Congress on Evolutionary Computation, pp. 222–227 (2005)
9. McMinn, P.: Search-Based Software Testing: Past, Present and Future. In: 4th International Workshop on Search-Based Software Testing, pp. 153–163 (2011)
10. McMinn, P.: Search-based software test data generation: A survey. Software Testing, Verification and Reliability 14(2), 105–156 (2004)
11. McMinn, P., Harman, M., Lakhotia, K., Hassoun, Y., Wegener, J.: Input Domain Reduction through Irrelevant Variable Removal and Its Effect on Local, Global, and Hybrid Search-Based Structural Test Data Generation. IEEE Trans. on Software Engineering 38(2), 453–477 (2012)
12. Nguyen, C.D.: Web page, tools, `http://selab.fbk.eu/dnguyen/public/cleaner-agent.tgz`
13. Nguyen, C.D., Miles, S., Perini, A., Tonella, P., Harman, M., Luck, M.: Evolutionary Testing of Autonomous Software Agents. Autonomous Agents and Multi-Agent Systems 25(2), 260–283 (2012)
14. Nunez, M., Rodriguez, I., Rubio, F.: Specification and Testing of Autonomous Agents in E-Commerce Systems. Software Testing, Verification and Reliability 15(4), 211–233 (2005)