Test-Plan Optimization for Flying-Probes In-Circuit Testers

Test-Plan Optimization for Flying-Probes In-Circuit Testers

Luciano Bonaria
SPEA Test Equipment
luciano.bonaria@spea.com

Maurizio Raganato
SPEA Test Equipment
maurizio.raganato@spea.com

Giovanni Squillero
Politecnico di Torino
giovanni.squillero@polito.it

Matteo Sonza Reorda
Politecnico di Torino
matteo.sonzareorda@polito.it

***Abstract*—The test of a printed-circuit board assembly often includes in-circuit test, which mainly aims at checking whether the different components have been correctly soldered. A tester may adopt either the bed of nails, or the flying-probes architecture. In the latter case, probes move to contact test points on each side of the board in order to perform the required tests. In order to minimize the test time, the sequence of movements of the probes should be re-arranged, considering the tester capabilities, the board layout, and several constraints coming from the environment and the customer. In this paper we describe the approach developed for optimizing tests on the SPEA 4080, which combines reduced test time with short test-generation time. Experimental results show the effectiveness of the proposed solution.**

Keywords— Test, ICT, Flying probes, Optimization

I. INTRODUCTION

Manufacturing Printed Circuit Boards (PCBs) is a complex process which requires extensive testing to guarantee acceptable levels of quality. Several test phases are typically implemented, often starting already at intermediate steps during the manufacturing process [1]. The checks performed by all these phases should guarantee that the test process is able to detect all targeted defects. Defects may concern each single component mounted on the board, the bare board, as well as the result of the assembly process.

The final test focuses on the defects introduced during the assembly. A common approach to assess the quality of the test is PCOLA/SOQ, that enable defining the defect spectrum for PCB Assembly (PCBA) manufacturing, allowing the test engineer to calculate a score for defect coverage [4][6]. Extended by iNEMI with PCOLA/SOQ/FAM, it represents the de-facto standard for computing the quality of the test of a PCBA process. Different test steps may contribute to such a metric, including Automated Optical Inspection (AOI), Automated X-ray Inspection (AXI), and In-Circuit Test (ICT). All these test steps aim at checking whether all components have been correctly mounted and soldered. For some components (mainly passive ones) an electrical test about their correct behavior can also be performed. However, a key goal is to check whether soldering has been correctly performed, and all connections work as expected.

Boundary Scan can support some of these techniques, when compliant components are adopted, but, in this scenario, a key role is played by ICT, which really checks the electrical properties of the target connections by applying/reading voltages and currents.

In-Circuit Test is implemented by resorting to probes which contact specific points on the boards (*test points*) and perform the required tests. The combined usage of AOI and ICT has also been explored [10]. With the emergence of new kinds of device packages (e.g., BGA), some points on the boards could not be accessed any more, and ICT can be complemented by Boundary Scan (BS). However, ICT continues to play a major role in PCBA testing, due to the huge number of connections which can be accessed by the probes and are not controllable/observable via BS (e.g., because they are related to passive or non-BS components). In-Circuit Test equipment are divided in two groups

- *Bed-of-nails*, where the number of probes (also called needles) corresponds to the number of test points. In this case, many tests can be performed in parallel, speeding up the whole test process. On the other side, for each PCBA a specific fixture mounting all the needles in the correct position is required, and sophisticated mechanisms are required to guarantee their correct positioning and contact.
- *Flying-probes*, where the number of probes is limited (up to 8 in high-end models) and they quickly move to contact the different test points, applying/observing the required electrical variable in a coordinated manner. The main advantage of this kind of equipment lies in its flexibility and in the lack of the fixture, at the cost of an increased test time.

In the last years, flying-probes testers improved significantly, increasing the number of probes, the speed and precision of their movements, while reducing their size, thus allowing to exactly contact very small test points. This trend allowed to reduce the comparative benefits of bed-of-nails and made flying-probes testers interesting even for mass production of complex PCBAs.

To support this trend significant investments have been made by the tester companies to improve the mechanics and the architecture of their products, such as the SPEA 4080 [5]. Such a tester is equipped with 8 probes moving on the two sides of the board under test

at unprecedented speed (up to 160 touches per second), and able to contact extremely small pads (down to 50μm). Special care has been devoted to minimizing the impact of vibrations due to the probe movements and of the related axis, as well as to guarantee the maximum level of precision in their movements. The introduction of this new generation of In-Circuit Testers may allow to widely extend the range of application of the flying-probe testers to PCBA mass-production, overcoming the limitations of the competing bed-of-nail technology.

Clearly, the duration of the whole test can be minimized not only by increasing the number of probes working in parallel and the speed of their movements, but also working on the minimization of the list of tests to be performed to achieve the target quality [2]. Moreover, the order according to which tests are performed significantly affects the total duration of the ICT test process, since it affects the movements of the probes from one test point to another and consequently the time they spend to move. Some early work on this problem was reported in [3]. However, the complexity of the PCBAs considered in that paper is very far from the one of current products, which easily include hundreds of components and tens of thousands of test points. Hence, traditional branch-and-bound algorithms can hardly be exploited, since they are not able to scale with the size of the problem at hand. Moreover, the optimization of the probe movements require taking into account not only the minimization of the path to be followed, but also a number of other constraints, such as those coming from the fact that multiple probes may be on the fly at a given time and any contact between them must be avoided, or those related to the pressure they create on the two sides of the board, or due to the presence of special components over which the probes cannot fly (*no-fly zones*).

This paper details the techniques developed for optimizing the flying probes movements in the new SPEA 4080 test equipment, a shorter description of the approach have been first drafted in [7]. The SPEA 4080 features eight high-speed probes, four on the top side and four on the bottom one, and it is used to touch 50μm pads on boards with a high density of components. Existing algorithms were unable to cope with the increasing dimension of the problem and to fully exploit the potential of the device.

The new algorithm is based on a dynamic greedy procedure that selects the optimal sequence of tests. In each step, the set of tests that will be performed in the subsequent measure is incrementally built by adding the one that would introduce the smallest delay given the current position of the probes. However, all the alternative probe positionings compatible with the tests to be performed are considered concurrently, as a set of possible implementations of the selected tests. The size of such a set increases in the beginning of the search process and slowly shrinks down to a single element as adding new tests increases the constraints, reducing the degree of freedom.

The greedy algorithm guarantees a linear number of steps on the number of required tests, while considering a set of alternatives in concurrently enhances the explorations and helps avoiding local minima. Moreover, the complexity of each step is bounded by a user parameter and does not depend on the size of the problem. Thanks to the adoption of new C++ 2011 standard[1], the algorithm can fully exploit the hardware resources of the SPEA 4080 and could be easily parallelized.

A summary of the results obtained by the proposed algorithm have been reported in [7]. This paper describes it in detail and includes full and updated results, showing some experimental results gathered on selected and representative PCBAs. Experiments demonstrate that the proposed solution significantly outperforms the previous ones, in terms of both generation time and test execution time.

II. BACKGROUND

A single ICT test t_i is performed by contacting a set of n test points $\overline{\boldsymbol{t}_i} = \{p_0^i, p_1^i, \ldots, p_{n-1}^i\}$, with $n = 2$ or $n = 3$ in most practical cases (although it may also be four in some cases).

A very simple example of a test with n=2 is the one involving a resistor: in this case two test points should exist at the two resistor terminals, and the test aiming at checking whether the correct component has been mounted may be based on either forcing a voltage drop across the resistor while measuring the current, or on forcing a current to flow through it, and measure the voltage drop. A simple example of a test with n=3 is when we have three resistors connected in parallel (see Fig. I). In order to measure the value of R1, one way is to use three probes, touching the test points **A**, **B** and **C**, respectively. The probes touching **B** and **C** force the voltage of these points to ground, while the probe connected to **A** and **B** can perform the test of R1, e.g., by forcing a known voltage on A and measuring the current flowing through B, knowing that no current will flow through R3 (*guarding*).

A *test block* T_j is a sequence of m tests $T_j = \left(t_0^j, t_1^j, \ldots, t_{m-1}^j\right)$ that must be performed in a precise order and with no interruptions. Most of the test blocks specify tests that must be performed on the very same component. For instance, testing a single transistor requires four tests, or five, in special cases; such tests are grouped into a single test block. Similarly, the 4 required to test an opto-isolator. The union of the test points that need to be contacted to perform all the tests in test block T_j is denoted with $\overline{\boldsymbol{T}_j} = \bigcup_{t_i} \overline{\boldsymbol{t}_i}$.

[1] ISO/IEC 14882:2011

Given a board, a dedicated software (called Test Generator in Fig. II) determines the *Test Plan*, that is, the set of all tests required to check that the board has been correctly assembled and that it is fully working—such a Test Plan achieves the appropriate level of confidence according to the PCOLA/SOQ standard. In order to generate the initial test Plan, information about the components (Bill of Material, or BOM) and layout of the board are required.

When the test plan is executed, the probes fly over and under the board to contact the required test points. The number of tests that need to be performed is fixed, but, while the order of tests inside a test block cannot be modified, the order of the test blocks themselves might be—even though with some constraints. For instances, all tests requiring the board to be powered must be performed together.

The goal of the optimization process implemented by the Test Optimizer is to reorder the test blocks inside the test plan, finding a sequence that minimizes the time required to execute the Final Test Plan. The Test Optimizer requires the original test plan, as well as the characteristics and constraints from the tester, like the number of probes, or their actual encumbrance or their acceleration and cruising speed.

To a first approximation, it may be maintained that the duration of the test plan strongly depends on the time required by the movements of the probes on the *x-y* plane. The problem faced by the optimization algorithm is to therefore to minimize the path of a set of probes, and may share some aspects with other path-length minimization problems. There are, however, important details that cannot be overlooked.

First, each step starts with all probes moving away from the board surface along the *z* axis, then the new locations are reached on the *x-y* plane, and eventually all probes simultaneously close on to contact the new test points moving along the *z* axis again. The time required to reach the new position does not depend linearly on the distance, as the probes accelerate and decelerate, and due to the mechanics, the performances can be different on the different axis. Anyhow, the eventual duration of the step is bounded by the slowest probe to reach its new position, that is, it is only relevant to optimize the slowest displacement among all probes.

Then, the number of tests that can be performed in a single positioning is variable: all probes required to perform a test may have already been specified by some other tests, thus, the extra test may come *for free*. For instance, if to perform tests t_x with $\overline{\boldsymbol{t_x}} = \{p^a, p^b\}$, and t_y with $\overline{\boldsymbol{t_y}} = \{p^a, p^c, p^d\}$, cumulatively four test points need to be contacted $\{p^a, p^b, p^c, p^d\}$, then test t_z with $\overline{\boldsymbol{t_z}} = \{p^a, p^c\}$ could be performed without additional movements. As a consequence, a careful order of the tests may lead to reducing the total number of displacements.

Moreover, the duration of the step is not only caused by the movements on the *x-y* plane: as all vertical movements are synchronous, if one probe needs to fly over a protruding component, all other probes will need to wait as well, slowing down the whole step. Differently, exceptionally protruding features may require a probe to find a circumnavigating path, slowing down the positioning even more (*no-fly zones* constraints). Finally, some test point may require an exceptionally slow movement to be contacted, regardless the actual path.

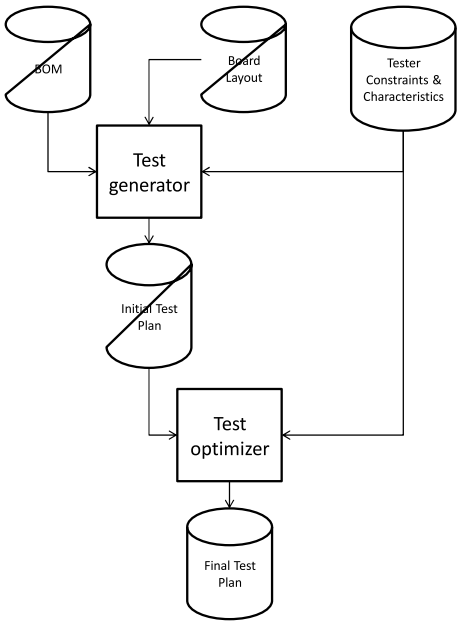

FIGURE I: OPTIMIZED TEST PLAN GENERATION

Clearly, the ability to correctly estimate the time required by each probe movement is a crucial issue in any optimization algorithm. This issue can be successfully faced only by accessing to the detailed information about the ATE mechanics.

Some type of test may require the use of special tools, such as an electroscan to detect opens, a laser, a light sensor, or an RFID tester. As a result, the duration of the step needs to include the time to switch the tool, and possibly the displacement required to fetch the tool itself if not already mounted on the head. Moreover, such tools are bulky, and one head may therefore limit the movement of other ones.

Finally, contacting some test points on one side of the board may require pushing the other side with a special *contrast* head.

Compared to a traditional minimum-path problem, flying probes impose several dynamic constraints: the obstructions caused by bulky tools are an example, the need for contrast is another. Moreover, the arms where the probes are mounted cannot cross: the positions on the x axis of the four probes on each side are required to be non-decreasing ($x_{p0} \geq x_{p1} \geq x_{p2} \geq x_{p3}$). Specific features of the board might make some test points not reachable by some probes.

At the same time, the flying probes scenario also increases the freedom for the solver: different test points may be connected to the very same net, thus, contacting them might be equivalent for performing certain type of tests. That is, the set of test points that could be contacted to perform a test t_i might be different from the one originally $\overline{\boldsymbol{t}_i}$ specified in the test plan. Indeed, these substitutions may not be always feasible, or the customer might have specific reasons not to do them.

Finally, it is worth mentioning that further constraints may be included in special cases. For example, in some cases it is important to take into account the need for coordinating the movements of the probes on the two side of PCBA, thus minimizing its possible bending during the test. Hence, the optimization algorithm must be flexible enough to accommodate for further requirements and constraints.

The boards tested with the new SPEA4080 easily contains thousands of test points and test plans, an order of magnitude more tests. Brute force, branch and bound exhaustive approaches are not applicable. On the other hand, the interdependency of the constraints complicates when not precludes the use of local heuristics, like the *k*-opts exploited by Lin-Kernighan algorithm [9]. More generally, the size of the search space calls for an efficient algorithm, but also the quality of the result is relevant — the customer cannot be left waiting neither when the test plan is optimized, nor when it is executed.

III. Optimization algorithm

Let P be a positioning a vector describing all eight flying probes: $P = (q_0, q_1, \dots, q_7)$. Each element in the tuple may specify either a test point on the board or the symbol $\boldsymbol{\varepsilon}$ if the position of the relevant probe has not been defined, yet. A *complete* positioning is a positioning that contains no $\boldsymbol{\varepsilon}$. A test block t is compatible with a positioning P, either complete or incomplete, if each test point it specifies also appears in the positioning $\forall t \in \overline{\boldsymbol{T}}: t \in P$.

A test block might be compatible with a set $\mathcal{P}$ of different incomplete positionings. For instance, let consider a test block that requires to contact three test points on the top side of a board t_a, t_b and t_c, the relative positions on the x axis are $x_a \geq x_b \geq x_c$, then it can be performed with four different positionings (t_a, t_b, t_c, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$), (t_a, t_b, $\boldsymbol{\varepsilon}$, t_c, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$), (t_a, $\boldsymbol{\varepsilon}$, t_b, t_c, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$), and ($\boldsymbol{\varepsilon}$, t_a, t_b, t_c, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}$), unless there are additional constraints.

The proposed algorithm (see Fig. III) finds the optimal sequence of tests by also defining a sequence of complete positionings. Starting from a list of tests $\mathcal{A}$, the procedure build the optimized list $\mathcal{T}$. In each step, given the *i*-th complete positioning P_i, it defines the next complete one P_{i+1} by repeatedly picking the test block that looks more promising, until no more tests may be selected.

In more details, the set $\mathcal{P}$ of potential future positioning initially contains only the completely unspecified positioning $\mathcal{P} = \{(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon})\}$, that is, a positioning compatible with the empty set of tests. As new tests are picked, $\mathcal{P}$ is updated to the set of all alternative positionings compatible with all selected tests. Therefore, the size of $\mathcal{P}$ increases in the beginning and then starts shrinking down, as adding more tests reduce the degree of freedom. Eventually, the function returns the best positioning compatible with all the selected tests, and it will be used as P_{i+1}. Indeed, when further tests cannot be added, $\mathcal{P}$ is likely to contain a single, complete positioning, and there is no alternative to select from.

```
procedure test_optimizer(𝒜):
    𝒯 = ( )
    P = initial_probe_position()
    while 𝒜 ≠ ∅ do
        𝒫 = compatible_positioning(∅)
        // 𝒫 = { (ε, ε, ε, ε, ε, ε, ε, ε) }
        x = best_test_block(𝒜, 𝒫, P)
        while x ≠ ε do
            remove x from set 𝒜
            append x to list 𝒯
            𝒫 = compatible_positioning(𝒯)
            x = best_test_block(𝒜, 𝒫, P)
        done
        P = best_alternative(𝒫)
    done
    return 𝒯
```

Figure II: Test Optimizer algorithm

In the step, tests are selected according to **`best_test_block`**, a proprietary heuristic that dynamically ranks the test blocks not yet included in the test plan according to the current position of the probes, their probable next position, and other structural information. Such a function is also delegated to handle most constraints imposed by the current scenario: physical limitations lead to considering test points more distant than they actually are, while the priority of tests containing test points that the customers prefer not to contact is lowered. As such heuristic is quite complex, it is the main point of trade-off between quality of the results and computational requirements.

As the exact timings required to reposition a flying probe depends on many different factors. Probes' acceleration and deceleration needs to be considered, and the speeds along the two axes are different. The algorithm exploits a mix of machine-learning techniques and practical heuristics to foresee the time required for each movement. First, the real times recorded for movements are analyzed using a symbolic-regression algorithm [8], and a polynomial function is generated. Then, the polynomial function is patched to take into consideration specific board features, such as no-fly zones, or the change of tools.

Additional constraints are handled by the function `compatible_positioning` that prune the positionings that violates physical constraints or are unacceptable by the customers. The resulting algorithm is able to handle all known constraints and can be easily extended to new ones by transforming them in test block priorities.

Finding the optimum solution of the addressed problem is probably NP-complete, since it corresponds to a modified Shortest Path Problem in the space of all possible positioning, that is $|\mathcal{P}| = \binom{number\ of\ test\ points}{number\ of\ probes}$. Moreover, in such a space the triangular inequality may not hold, as distances encode the time used to move all probes from one positioning to the next. Given the size of the boards addressed in practice, only heuristic approaches can be considered, such as the one proposed in this paper.

The algorithm is inherently greedy, but weights are updated dynamically, and by considering the set of all potential future positioning $\mathcal{P}$ in each step it effectively explores the search space and avoids getting stuck in local optima.

Discussing the complexity of the algorithm, the number of steps of the optimization process increases linearly with the number of tests. Anyhow, during the optimization, all tests need to be selected; for each of them, the weight of all the remaining ones must be recalculated, and the heuristic adopted needs to scan all test points on the board. Let n_t be the number of tests and n_p be the number of test points, the theoretical complexity of the procedure may therefore be approximated as $\mathcal{O}(n_t^2 n_p)$. It must be pointed out that, when large boards are considered, the heuristic procedure only considers a subset of test points.

However, as in many industrial problems, the asymptotic complexity is interesting only as a mean to reduce the computational time experienced by the customer. The algorithm is designed to let its implementation exploit a modern C++ compiler. Namely, the introduction of *rvalue references* and the new standard template library introduced in 2011 allowed quite significant optimizations. Moreover, the memory layout was optimized to take full advantage of the CPU prefetcher and the 64-bit architecture.

IV. Experimental analysis

The algorithm was implemented in about 5,000 lines in C++, following the standard ISO/IEC 14882:2017[2]. For the sake of comparison, Table II reports the performances of FP2012, the algorithm previously used on SPEA ATEs, when executed on the new SPEA 4080. Comparison with algorithms implemented on other testers is unfeasible, since results strongly depend on the characteristics of the tester itself.

It should be noted, however, that FP2012 was designed and optimized for the type of boards tested on the machines of the previous generation, and its performances are sub-optimal when the number of tests and test points is high. Moreover, the older algorithm did not target the hardware available on the SPEA 4080.

TABLE I. Board Characteristics

Board	Physical		Test plan
	Size [cm]	Test points	Tests
Board 1	25.59 x 18.30	134	1,252
Board 2	24.89 x 23.63	143	591
Board 3	11.95 x 13.53	167	855
Board 4	19.14 x 16.11	360	2,003
Board 5	14.51 x 13.94	528	2,791
Board 6	28.44 x 15.28	680	5,931
Board 7	27.46 x 25.41	704	1,915
Board 8	20.67 x 11.00	950	64,428
Board 9	31.33 x 27.86	996	4,022
Board 10	35.36 x 23.79	1,849	17,820
Board 11	12.24 x 13.07	2,267	12,317
Board 12	49.18 x 46.47	11,950	48,759

Results are reported in Table II on a dozen of boards where the SPEA 4080 is used, with different characteristics in terms of test points and tests, and different dimensions. Some boards are internal testbenches, while other correspond to real products. The comparison takes into account both the computational time required to optimize the test plan (Opt) and the time required to execute it (Exe).

FP2018 is more efficient optimizing the test plan, with an average increment in performance of 80%, and a peak of 98% on Board 11, where an optimization time of 11 minutes is reduced to a mere 13 seconds. It is important to underline that the optimization time remains within a reasonable range of values (i.e., minutes) even for the largest and most complex boards.

The test plans optimized by the new algorithm are also more effective and require less time to be executed. Improvements range from a 7% on Board 4

[2] https://www.iso.org/standard/68564.html

(from 29 seconds to 27), to a 65% on Board 3 (from 23 second to 8), with an average reduction of 32%.

In Fig. IV we graphically reported the improvements provided by the proposed algorithm in terms of both optimization and execution time.

TABLE II. TIME REQUIRED FOR THE OPTIMIZATION AND THE EXECUTION OF THE TEST PLAN ON THE SPEA 4080

Board	FP2012		FP2018	
	Opt [s]	Exe [s]	Opt [s]	Exe [s]
Board 1	12	12	1	10
Board 2	9	2.5	1	2
Board 3	52	23	3	8
Board 4	119	29	6	27
Board 5	61	45	13	28
Board 6	70	58	16	29
Board 7	55	32	28	24
Board 8	1,890	1,348	308	1,056
Board 9	72	42	20	36
Board 10	1,167	168	194	133
Board 11	660	244	13	88
Board 12	510	819	320	489

Interestingly, the performance of the proposed algorithm does not seem to degrade with either the number of test points or tests.

V. CONCLUSIONS

This paper describes the characteristics and performance of the new SPEA 4080 flying-probes In-Circuit test generation and optimization steps, with special emphasis on how to identify the optimal sequence for the test to be performed.

The architecture of the system is described, the many constraints existing when testing real PCBs are summarized, and an optimization algorithm is outlined which is able to take all of them into account. Experimental results gathered on a set of different boards show that the proposed solution is able to take full advantage of the new architecture of the SPEA 4080 system, and in particular of the available 8 probes. The computational time required to optimize the test plan remains within an acceptable range of values, while the execution time is significantly reduced, thus widening the set of scenarios where flying-probes ICTs can be effectively adopted. The approach is flexible, as it allows the test engineer to freely set the values of a few parameters, thus trading off between the computational time required by the optimization phase, and the duration of the optimized test plan.

FP2018 was designed to exploit a multicore architecture. While the current version of the algorithm is not concurrent, a multithreading version of the algorithm is being developed. In this way we plan to further reduce the optimization time, thus allowing to manage even more complex boards, and/or to reduce even further the test execution time.

Hence, we claim that our work represents a significant step ahead towards making the flying probe systems economically suitable even for PCBA mass production test.

ACKNOWLEDGMENT

Authors wish to thank Michelangelo Di Fronso and Domenico Pirilli for sharing their experience on flying-probes testers and their support in fine tuning the algorithm; Andrea Firrincieli, for implementing the first version of the optimizer, and Danilo Dimiccoli for completing the work.

REFERENCES

[1] C.F. Coombs and H.T. Holden. Printed circuits handbook; McGraw-Hill, 7th edition, 2016

[2] Harm van Schaaijk, Martien Spierings, Erik Jan Marinissen. “Automatic Generation of In-Circuit Tests for Board Assembly Defects”; 2018 IEEE International Test Conference in Asia (ITC-Asia)

[3] Yuki Hiratsuka, Fumihiko Katoh, Katsumi Konishi, Seiichi Shin. “A design method for minimum cost path of flying probe in-circuit testers”; Proceedings of SICE Annual Conference 2010

[4] T. Taylor. “Functional Test Coverage Assessment Project”; IEEE 8th International Board Test Workshop (BTW’2009), Fort Collins, USA, Sept 15-17, 2009

[5] http://www.spea.com/BoardTestAutomation/ProductsbyFunction/4080/tabid/427/language/en-US/Default.aspx

[6] K. Hird; K. P. Parker; and B. Follis. “Test coverage: what does it mean when a board test passes?”, IEEE International Test Conference (ITC), pp. 1066–1074, 2002.

[7] L. Bonaria; M. Raganato; M. Sonza Reorda; G. Squillero. “A Dynamic Greedy Test Scheduler for Optimizing Probe Motion in In-Circuit Testers”, European Test Symposium (ETS), 2019

[8] J. Koza, M. Keane, J. Rice. “Performance improvement of machine learning via automatic discovery of facilitating functions as applied to a problem of symbolic system identification”, IEEE International Conference on Neural Networks, 1993, pp. 191–198.

[9] K. Helsgaun. “General k-opt submoves for the Lin–Kernighan TSP heuristic”, Mathematical Programming Computation, 2009, Vol 1, Is 2–3, pp. 119–163

[10] P. Radev, M. Shirvaikar. “Enhancement of flying probe tester systems with automated optical inspection”, 2006 Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory, pp. 367 – 371