A Novel Technique for Generation of Test Cases Based on Bee Colony Optimization and Modified Genetic Algorithm (BCO-mGA)

Sandeep Dalal
Assistant Professor,
Department of Computer Science and Applications, M.D.University, Rohtak, (Hr)-India-124001

Rajender Singh Chhillar, PhD.
Professor & Head,
Department Computer Science and Applications, M.D.University, Rohtak
(Hr)-India-124001

ABSTRACT

Software testing is the most important phase of software development life cycle which ensures the quality of software systems. This paper proposes a novel approach for generation of test cases from "Unified Modeling Language" (UML). The Test Case Selection and Reduction is done by using Stratified Sampling, Bee Colony Optimization and Genetic Algorithm. This aims to save cost, time and effort by efficiently minimizing the test suite to ensure maximum coverage. We have applied this technique to a module of a Card Administration System project taken from Software Company. The result shows that the proposed approach effectively detects all the flaws by covering all possible paths of the system. The proposed software testing technique (BCO-mGA) ensures maximum coverage in minimum possible timeframe by executing the final minimized test suite.

General Terms:

Algorithm, Design, Standardization and Verification

Keywords:

Software Testing, Test Cases, UML, Activity Diagram, Activity Diagram, Genetic Algorithm, Stratified Sampling, Bee Colony Optimization, Cross-over, Progeny

1. INTRODUCTION

Software testing is a process that is used to validate and verify the developed program to ensure that it satisfies the customers and developers requirements [1] [2]. Now industry is in mature stage and lots of upgrades are done to the current projects as part of maintenance. Based on the previous study of root cause analysis of some of the most critical failure, [3] [4] [7] it is observed that lots of software fail due to lack of testing. Quality of the delivered software depends heavily on the methodical action of software. Software testing plays a crucial role in assuring software quality. It can be used for the purposes of quality assurance, reliability, estimation, verification and validation [5]. Software testing is difficult due to large number of test cases. Test Case Execution is cost and investment intensive. We propose a novel technique for generation and reducing test cases, based on bee colony optimization and modified genetic algorithm (BCOmGA). Our testing strategy derives test cases using full predicate coverage criteria.

1.1 Stratified Sampling

A method of sampling that involves the division of population into smaller groups known as strata. In stratified sampling the strata are formed based on members shared attributes or characteristics [26]. A sample for each stratum is taken in a number proportional to the spectrums size when compared to the size of population. These subsets of strata are then pooled to form a random sample. Stratified sampling ensures that atleast one observation is picked from each of the strata, even if the probability of it being selected is far less than one [27].

1.2 Bee Colony Optimization

Bee colony Optimization is an emerging field for researchers. Bee colony optimization is the name given to the colony formed from the mutual understanding and terms work of the natural bees in the process of foraging [6]. The bee colony optimization has been used for understanding the concept of software test suite optimization [8]. The bee's concept has also been used for optimization approaches in the field of engineering. In spite all the creatures on the earth follow one or the other mechanism to find food that suite them and these mechanism are found in insects like ants, bees and cockroaches etc. There are three types of bees queen bees, male drone bees, workers bees. Queen bee (one) is responsible to lay eggs which are used to build new colonies. Male drone bees are responsible for mating with queen bee. Worker bees (thousands) perform all the maintenance and management jobs in the hive. Honey bee comb build-up and management is a classic example of team work, co-ordination and synchronization. The way the honey bees find, build and maintain their comb and hive is remarkable. These are the factors which have given rise to interest of researchers to find solutions to their problem.

1.3 Genetic Algorithms

GA is adaptive search procedures which were introduced by John Holland [13] and widely studied by Goldberg [14].Genetic algorithm is well suited to identify UML testing. Genetic algorithms work on the basis of combination of 'genes', which are created randomly; followed by a directed evolution phase wherein the "combination of genes" evolve under a desired selection pressure. High scoring combinations are kept for next phases, while others are discarded. In short, genetic algorithms optimize test parameters that satisfy a predefined test criterion.

In this paper we design new test cases and testing techniques that are based on root cause analysis of software failure. In Section 2, we discuss related work and research in field of software testing to optimize test cases. Section 3 describes proposed model. Section 4 presents proposed algorithm prioritization and maximizing coverage, while in Section 5 discussion about reduction of nodular path. Some future directions have been presented in section 6.

2. RELATED WORKS

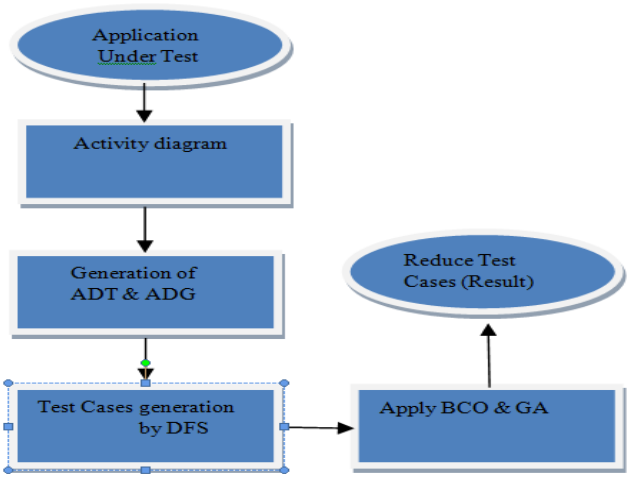
In this section we describe the number of approaches which have been proposed for test case optimization. UML is the most widely used language; many researchers are using UML diagrams such as state chart, Activity and sequence diagrams to generate test cases [12]. P.R. Srivastava and T.H.Kim [24] proposed a technique for generating test cases using 'path coverage testing criteria' and genetic algorithm many researchers have been working in generating optimal test cases based on specification. Boghdady et al [11] proposed an approach for generating test cases from activity diagram. The proposed model includes validation of generated test cases during generation process to ensure their coverage and efficiency. Automatic test case generation was proposed by Monalisa et al [15] from UML design diagrams. The concept of Artificial Bee Colony algorithm was introduced by Karaboga [16] [20].Genetic algorithm (GA) is an optimization technique which can be applied to various problems, including those that are NP-hard [22]. With this the genetic algorithm is also used to generate test data automatically [17]. A lot of work is done by researchers on optimization of test cases. Mala et al has developed a hybrid genetic algorithm based approach for quality improvement and optimization of test cases [19]. Dahiya et al [23] presented an ABC algorithm

based approach for automatic generation of structural software tests. Bharati et al [25] have used a combination of BCOGA for a test case generation; however the method of determination of estimated time is not clear in her paper whereas in this paper we have based our approach on fitness score (which is a combination of "Information flow" and "Stack based weight").

3. PROPOSED MODEL

This proposed model studies the Activity diagrams of UML as a building block to reduce the number of nodular paths to be tested for the validation of the software. From the Activity diagrams a Activity dependency table (ADT) is generated and Activity dependency graph (ADG) is created. By the application of depth first search algorithm (DFS) we have generated the test cases. On the basis of the generated test cases stratified sampling, bee colony optimization and modified genetic algorithm is applied to reduce the number of nodular paths to be tested for the validation of the software.

Figure 1: The Proposed Model Architecture



4. PROPOSED ALGORITHM

The proposed approach reduces the nodular path options by a combined application of bee colony optimization and modified genetic algorithm. Our proposed algorithm involves the following steps.

Assumption: AD and AG will be generated for each module.

START

INPUT: AUT (Application under test)

Test Data: B (B1, B2 ----------Bn)

Variables: C (C1, C2----------Cn)

STEP 1:- Generate Activity Diagram (AD) from AUT

- Generate ADG (Activity Dependency Graph) from AD

- Calculate total set of employee class by stratified sampling

STEP 2: Generate and evaluate Nodular Path (NP)

- Use depth first search to traverse the graph following each node
  /*Application of BCO-mGA*/
- Get all Paths (test cases) $P[\ ] = P_1, P_2 \ldots\ldots P_n$.
- Randomly Select "m" Employees Classes(ECMs) for initial foraging
- Each Employee Class Member (ECM) will randomly start foraging
- Calculate the number of Decision nodes covered (DNC) and Fitness score (FS) on the basis of Path foraged
- The Paths of returning ECM will be stored in memory
- Second Path would be randomly assigned to the ECM
- DNC and FS calculated for new Path Combination
- Best Fit solutions would be crossed
- Select progeny with Path combinations of cross over having higher FS than parents
- The new ECM would be assigned to the Progeny Path
- The steps of foraging, FS calculation and Crossover would be repeated till either
  - Number of fixed iterations
    OR
  - No more improvement of Progeny path

Figure 2: Activity diagram "Card Management system"

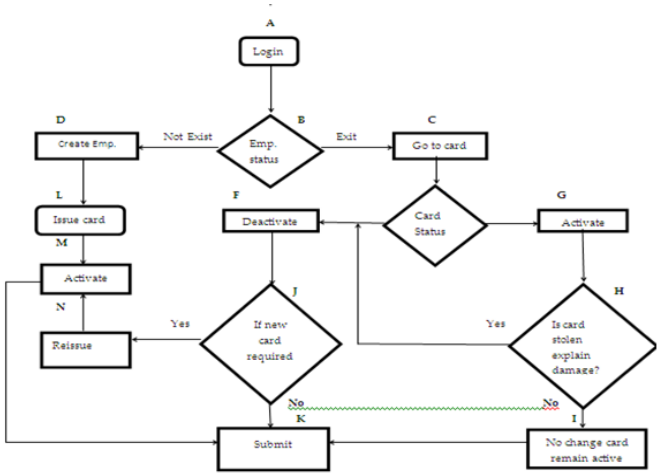
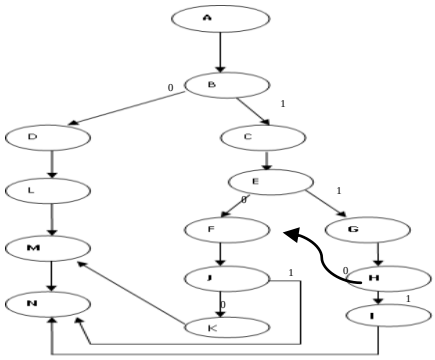


Figure 2 represents the Activity diagram of the card management system. On the basis of Activity diagram, the ADT is accomplished to automatically generate the Activity dependency graph. In figure 3 the symbols given for each activity are used to name the nodes in the ADG where each node represents an activity diagram.

Figure 3: Activity dependency graph



Depth First Search (DFS) is applied on the ADG to obtain all the possible test paths. The flow chart of the methodology is presented in figure 4. It can be said that the methodology will accept raw sequence set as the input and will generate the optimal test sequence. All the possible test sequences are needed to be generated to perform the given task.

Figure 4: Flow chart of Algorithm stack based

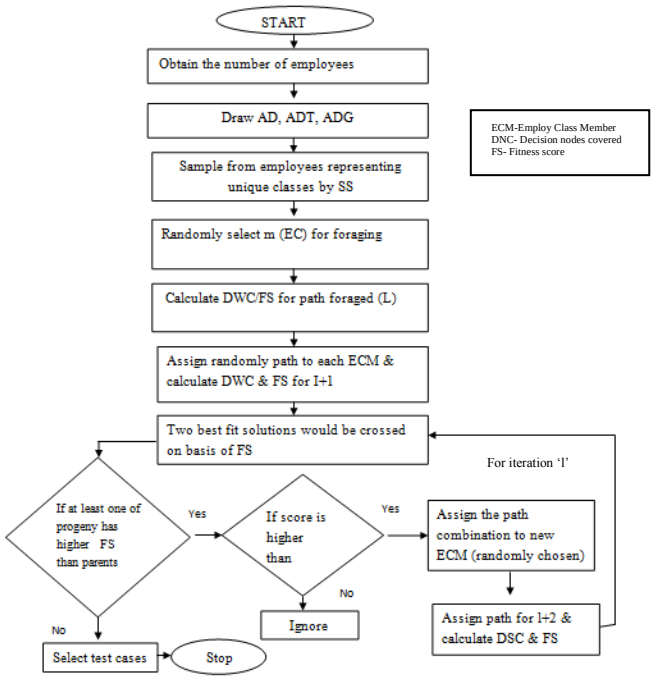


Table1: Node Weight table; Stack Based Weight (SBW_i) and K_i=node number (i)

SBW_i	$W_{max} - K_i$
1	14-1=13
2	14-2=12
3	14-3=11
4	14-4=10
5	14-5=9
6	14-6=8
7	14-7=7
8	14-8=6
9	14-9=5
10	14-10=4
11	14-11=3
12	14-12=2
13	14-13=1
14	14-14=0

Table2: Information Flow Table

IF	(Fanin(A) x Fanout(B))
2	1*2 = 2
6	2
11	2
10	2

Table 3: Cumulative Weight Table (Wi = IFi + SBWi)

Activity	Node Number (i)	IF	SBW_i	W_i
A	1	1	13	14
B	2	2	12	14
D	3	1	11	12
C	4	1	10	11
L	5	1	9	10
E	6	2	8	10
M	7	1	7	8
F	8	1	6	7
G	9	1	5	6
J	10	2	4	6
H	11	2	3	5
I	12	1	2	3
K	13	1	1	2
N	14	1	0	1

Table 4: Nodular Path Generation

S. No	Nodular Path Sequence
NP1	A=>B=>D=>L=>M=>N
NP2.	A=>B=>C=>E=>G=>H=>I=>N
NP3	A=>B=>C=>E=>G=>H=>F=>J=>N
NP4	A=>B=>C=>E=>G=>H=>F=>J=>K=>M=>N
NP5	A=>B=>C=>E=>F=>J=>N
NP6	A=>B=>C=>E=>F=>J=>K

Table 5: Over all fitness of all Nodular Paths

NP	DN1	DN2	DN3	DN4	Fitness
NP1	0	0	0	0	14
NP2	1	1	1	0	29
NP3	1	1	0	1	30
NP4	1	1	0	0	24
NP5	1	0	0	1	20
NP6	1	0	0	0	14

5. CASE STUDY

Problem Statement

The study deals with the efficient and comprehensive testing of various possibilities defined by the automated card management system. Consider the Activity diagram derived from the proposed algorithm; there are 6 unique nodular paths for each employee. If there are 'N' employees, then the total number of cases becomes 6N. For any organization with large number of employees it becomes a mammoth task to verify the performance of algorithm on each level. In the current case the application BCO-mGA steps. DFS is applied AD for generation of test cases that produced all possible paths. These will serve as input for the combination of BCO-mGA. As shown in table 4 there are six test cases [NP1, NP2……NP6]. The weight of each node (w_i) is dependent on the "Information Flow(IF)" and "Stack Based Weight(SBWi)" as shown in Table 3. Fitness of a nodular path depends upon its ability to unravel faults and therefore must be proportional to the number of decision nodes it covers.

As seen in the "Activity Dependency Graph(ASDG)" nodes B, E, H and J are decision nodes (IF > 1) with 14, 10, 5, 6 scores respectively. As seen in Table 5 the scores of nodular paths are [14, 29, 30, 24, 20, 14] for [NP1, NP2 ……NP6] respectively. Assuming the number of employee class after is 12 out of which only 4 (1/3) start foraging. In first phase each of these employee class is randomly assigned a test case (NPx;

where x =1...6) and fitness score (FS_{npx}) calculated as:

$$FS_{npx} = IF_x + SBW_x$$

Where, $IF_x = Fanin\text{A} * Fanout\text{A}$

Fitness Score = $^{N}\sum_{x=1} NP_x$ [where, N is the maximum test cases]

The numbers of decision nodes covered are calculated in binary form. In second phase, the employee class members (ECM) of phase 1 are randomly assigned a second path and FS_x calculated. The top scoring two ECM are selected as parents and crossed to produce new combinations of test cases (for example in the phase2 ECM3 and ECM4 are selected as parents). The numbers of decision nodes covered are calculated by using ADG. In case, the offspring test case combinations have higher fitness score than any of the parents, then the offspring is assigned to another randomly chosen ECM for foraging. Else next best set of parents are chosen and crossed. In third round, the ECMs are assigned another nodular path randomly and FS is calculated and the parents for another round of crossover selected. This process is repeated till one of the three conditions is satisfied:

a. The number of iterations is exhausted.
b. None of the offspring (for any set of parents) show higher fitness score. This is an indication that the optimum combination has been reached.
c. All of the decision nodes are covered.

Phase 1

EMP	NPS	FS	DNC	BIN
E1	NP1	14	0	0000
E2	NP2	29	3	1110
E3	NP3	30	3	1101
E4	NP4	24	2	1100

***** Nodular Path Sequences (NPS); * Decision Nodes Covered (DNC)
* Fitness Score (FS); * Binary representation of NPS (BIN)

Phase 2

EMP	NPS	FS	DNC	BIN
E1	NP1,NP5	34	2	1001
E2	NP2,NP6	43	4	1111
E3	NP3,NP1	44	3	1101
E4	NP4,NP5	44	4	1101

Crossover of Selected parents: from Phase 2

EMP	NPS	FS	DNC	BIN
E1	NP1,NP5	34	2	1001
E2	NP2,NP6	43	4	1111
E3	NP3,NP1	44	3	1101
E4	NP4,NP5	44	4	1101
E5	NP3,NP5	50	5	1101

Phase 2 (after cross over)

Parents	NPS combination	Progeny	Fitness score	Status
E3	NP3,NP1	NP3-NP5	50	Selected
E4	NP4,NP5	NP4-NP1	38	Rejected

Phase 3

EMP	NPS	FS	DNC	BIN
E1	NP1,NP5,NP4	58	4	1101
E2	NP2, NP6,NP3	73	7	1111
E3	NP3,NP1,NP5	64	5	1101
E4	NP4,NP5,NP2	73	7	1111
E5	NP3,NP5,NP1	64	5	1101

Crossover of Selected parents: from Phase 3

Parents	NPS combination	Progeny	Fitness score	Status
E2	NP2,NP6,NP3	NP2-NP5NP2	78	Selected
E4	NP4,NP5,NP2	NP4-NP6NP3	68	Rejected

The process stopped as all of the decision nodes were covered in the phase 3 (Emp2 and Emp4). These are the end result i.e. selected final outcome of NPS combination that would cover entire set of decision nodes.

6. REDUCTION OF NODULAR PATHS

Usually the nodular paths are generated to cover maximum routes on an algorithm. In this case we have attempted to reduce the nodular paths with respect to the decision nodes of the Activity diagram. With each path a unique set of decision nodes are associated and hence convey different fitness weight. The nodular paths validate the putative routes that can be followed during the operation of a desired activity (under some constraints). The possible test cases are here in table 4**.** The test cases have been defined on the basis of state flow diagram. It is evident that the nodular paths can represent different options/choices before meeting final objective. The nodular paths can be generated through the analysis of state flow diagram. Some of the possible test sequences are defined *in table 4.*

The above nodular path sequences (NPS) are defined with effect of the nodes, here A, B, C etc represents the nodes. Let's say if we have a sequence A=>B=>D=>L=>M=>N. In physical terms it represents the activity sequence of login, employee status determination (whether it exists or not), create employee id (if it does not exist), activate its card, check if it is active, submit the information.

From all the possible nodular paths it can be observed that there is always one initial node and the related path verification represents the completion of movement on the path. Initially the login of the employee will trigger a decision by system: Whether the login exists or not? This will produce two outputs either yes or no. Any later movement on the path will be determined on the outcome of this initial node.

7. CONCLUSION

The proposed approach reduces the nodular path options by a combined application of bee colony optimization and modified genetic algorithm. The technique developed using this approach identifies and reduces the test data. This approach provides better results in the initial iteration of the complete process. It provides positive feedback and hence it can lead to superior solutions in optimum time. This technique is particularly focused on the decision nodes functioning which are major determinants of downstream information flow routes. Tools based on such approach will not only reduce the number of test cases for the comprehensive validation of software but also will lead to over all improve the quality of fault managements (specifically during testing phase; which accounts for more than 31% software failures) as per our previous findings of empirical study of root cause analysis of software failure. Future outlook based on current algorithm would be to develop another algorithm for automatic fault rectification.

8. REFERENCES

[1] Mathur Aditya P "*Foundations of Software Testing*" Pearson Education India, 2008

[2] Pressman, R.S. 1997. *Software Engineering: A Practitioner Approach*, 4th Edition, Tata McGraw Hill.

[3] Sandeep Dalal and Rajender Singh Chhillar, "*Software Testing Three P's Paradigm and Limitations,"* in International Journal of Computer Applications, vol. 54, no. 12, pp. 49-54, September 2012

[4] Sandeep Dalal and Rajender Singh Chhillar "*Case Studies of Most Common and Severe Types of Software System Failure"* International Journal of Advanced Research in Computer Science and Software Engineering Volume 2, Issue 8, August 2012.

[5] Myers, G.J. *The Art of Software Testing*, New York: John Wiley and Sons

[6] S.S.Dahiya, J.k.Chhabra, S.Kumar, "Application of Artificial Bee Colony Algorithm to Software Testing", Software Engineering Conference (ASWEC), 21st Australian IEEE Conferences,2010.

[7] Sandeep Dalal and Rajender Singh Chhillar "*Role of Fault Reporting in Existing Software Industry,"*CiiT International Journal of Software Engineering and Technology July 2012.

[8] D. Jeya Mala, V. Mohan, "ABC Tester -Artificial Bee Colony Based Software Test Suite Optimization Approach", Int.J. of Software Engineering, IJSE Vol.2 No.2 July 2009.

[9] Bertolino, A. and Basanieri, F. 2000. "A Practical approach to UML-based derivation of integration tests". In Proceeding of the Fourth International Software Quality Week Europe and International Internet Quality Week Europe(QWE), Brussels, Belgium.

[10] Swain, S.K. Mohapatra, D.P. and Mall, R. 2010. "Test Case Generation based on Activity and Sequence Diagram", *IJSE*, Vol. 3, 2010, 21-52

[11] Boghdady, P.N., Badr, N.L., Hashem, M. and Tolba, M.F. 2011. "A Proposed Test Case Generation Technique based on Activity Diagrams", *IJENS*, 11, 37-57

[12] Prasanna, M., Chandran, K.R. and Suberi, D.B. (2011): "Automatic Test Case Generation for UML Class Diagram using Data Flow Approach", *Academia.Education*

[13] J.Holland, "Adaption in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press,1975.

[14] D. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", New York,Addision Wesely, 1989.

[15] Sarma, M. and Mall, R. 2007. "Automatic Test Case Generation from UML Models", 10th International Conference on Information Technology, pp. 196-201.

[16] D. Karaboga, B. Basturk, "A Powerful And Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm", Journal of Global Optimization, Volume:39 , Issue:3 ,pp: 459-471, Springer Netherlands,2007

[17] Marc Roper, Iain Maclean, Andrew Brooks, James Miller and Murray Wood. Genetic Algorithms and the Automatic Generation of Test data,1995.

[18] W.W.Eric, ,R.H.Joseph, L.Saul and Aditya P.Mathur,"Effect of Test Case Minimization of Fault Detection Effectiveness",Software Practice and Experience,Vol.28,No.4, pp. 347- 369, 1998.

[19] D.J.Mala , V.Mohan, "Quality Improvement and Optimization of Test Cases-A Hybrid Genetic Algorithm Based Approach", ACM SIGSOFT ,May 2010

[20] D. Karaboga, B. Basturk Akay, "Artificial Bee Colony Algorithm on Training Artificial Neural Networks, Signal Processing and Communications Applications", .SIU 2007, IEEE 15th. 11–13 June 2007, Page(s):1 - 4, 2007.

[21] D. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", New York,Addision Wesely, 1989.

[22] NP-Hard Problems", ACM SIGACT, Volume 28 Issue 2, June 1997.

[23] S.S.Dahiya, J.k.Chhabra, S.Kumar, "Application of Artificial Bee Colony Algorithm to Software Testing", Software Engineering Conference (ASWEC), 21st Australian IEEE Conferences,2010

[24] P.R Srivastava,T.Kim,"*Application of Gentic Algorithm in Software Testing*", Internation journal of software engineering and its applications,Vol.3,no4,oct 2009,pp-87-96.

[25] Bharti Suri, Isha Mangal & Varun Srivastava" *Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm*" Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT) : 2231–5292, Vol.- II, Issue-1, 2

[26] SARNDAL, C.-E., SWENSSON, B., AND WRETMAN, J. 1992. Model Assisted Survey Sampling. Springer-Verlag, New York, NY.

[27] COCHRAN, W. G. 1977. Sampling Techniques. John Wiley & Sons, Inc., New York, NY

.