IDAP INSTITUTE OF DEFENSE ACQUISITION PROGRAM

JAMS JOURNAL OF ADVANCES IN MILITARY STUDIES

2025, Vol. 8, No. 1, 15-36.

https://doi.org/10.37944/jams.v8i1.275

Optimization based machine learning algorithms for software reliability growth models

Shin, Myeonggeun* · Jung, Juwon** · Lee, Jihyun*** · Ryu, Insoo**** · Park, Sanggun*****

ABSTRACT

Software reliability is a critical factor for system performance and safety, especially in defense industries, where operational failures can have severe consequences. To evaluate and improve software reliability, Software Reliability Growth Models (SRGMs) are widely used. However, many previous studies have relied on single optimization methods or deep learning approaches, which are prone to local optima and extrapolation issues, reducing prediction accuracy. To fill this gap, current study employs a broader range of optimization algorithms based on the Least Squares Method (LSM) and Maximum Likelihood Estimation (MLE) to approximate global optima. NASA's Jet Propulsion Laboratory (JPL) software defect datasets were used, and several widely recognized SRGM models, including Goel-Okumoto, Delayed S-Shape, Inflection S-Shape, Weibull, and Log-Logistic, were evaluated. Experimental results show that the choice of optimization method significantly affects prediction performance, as measured by Mean Squared Error (MSE). For example, in the J2 dataset, the Weibull model exhibited MSE values ranging from 70.778 to 15,767.68—a 222-fold difference—demonstrating the critical role of optimization in prediction accuracy. The findings confirm the risks of relying solely on single-method approaches and highlight the value of diverse optimization strategies for achieving near-global optima. The study presents a practical framework for improving software reliability assessments, contributing to the development of highly reliable software for the defense industry.

Keywords : software reliability, software reliability growth model, artificial intelligence optimization, machine learning

* (First author) MOASOFT Corp. and Korea University, Graduate School of Engineering & Technology, Research Engineer (Master's Degree Candidate), mgshin@moasoftware.co.kr, https://orcid.org/0009-0001-3351-4580.

** (Co-author) MOASOFT Corp. and Kwangwoon University, Department of Defense Industry AI & Robot Convergence, Research Engineer (Master's Degree Candidate), jwjung@moasoftware.co.kr, https://orcid.org/0009-0007-4192-2713.

*** (Co-author) MOASOFT Corp. and Kwangwoon University, Department of Defense Industry AI & Robot Convergence, Senior Research Engineer (Master's Degree Candidate), jhlee@moasoftware.co.kr, https://orcid.org/0000-0001-8165-2657.

**** (Co-author) MOASOFT Corp., AI/Data Science Lab., Principal Research Engineer, isryu@moasoftware.co.kr, https://orcid.org/0009-0002-3215-0102.

***** (Corresponding author) MOASOFT Corp. and Kwangwoon University, Department of Defense AI & Robot Convergence, Senior Research Engineer (Ph.D. Candidate), sgpark@moasoftware.co.kr, https://orcid.org/0009-0001-3196-510X.

Ⅰ. 서론

현대 사회에서 소프트웨어 운영 중 불능(malfunction)이나 문제(trouble) 발생은 시스템의 성능과 안전성에 상당한 영향을 미치게 된다(Wang & Zhang, 2025). 그래서 제품의 신뢰성 평가는 제품을 시장 출시 전에 잠재적인 문제를 조기 감지하고 실제 조건에서 성능 작동을 확인하는 과정(Samal, 2024)으로 필수적이다. 또한, 기존 버전의 업그레이드 과정에서도 변동에 따른 결함 문제가 발생할 수 있어 소프트웨어 품질과 안정성 차원에서 소프트웨어 신뢰성이 중요하다(Chatterjee, Saha, & Sharma, 2021). 이런 소프트웨어 신뢰성 보증의 중요성은 민간산업 뿐만 아니라 방위산업 무기체계 운용 차원에서 더욱 강조되고 있다. 최근 첨단 무기체계 획득요구와 상호운용성이 높아지면서 제품(무기체계)의 복잡성이 증대되고 있어 소프트웨어의 구현 비중이 증가하고 있다.

방위산업 분야의 무기체계에 적용된 소프트웨어는 활용 코드의 결함 요소(영향도, 발생빈도, 제어가능성)를 사전에 식별하고 위험성을 수정하는 목적의 소프트웨어 신뢰성 시험을 통과해야 한다(Oh et al., 2024). 게다가 최신 무기체계는 무인 체계(센싱, 네트워킹, 빅데이터, AI) 운용 확대를 추진하고 있어 체계의 고신뢰성이 중요하며(Kim, Kim, & Jeong, 2024), 소프트웨어 기반 기술의 발전으로 성능개량・기능 변경 요구가 증가하면서(Cho, Min, Lim, & Choi, 2023) 미래 전투준비태세와 작전지속능력 수준을 유지하는 지속적인 품질개선 활동이 필수적이다(e.g., Hur, 2024). 특히, 해당 체계나 장비는 장기간의 운용 보장을 위한 일련의 시험(test), 분석(analysis) 및 수정(fix) 과정인 신뢰도 성장(reliability growth) 개념을 토대로 신뢰성 수준을 점진적으로 개선하는 지속관리가 필요하다(Sung, Yoon, & Lim, 2021).

이런 지속관리 차원에서 소프트웨어 신뢰성 성장 모델(Software Reiability Growth Model, SRGM)은 신뢰성의 평가 및 개선을 위한 효과적인 대표적 방법으로 사용되고 있다(Bahnam, Dawwod, & Younis, 2024; Samal & Kumar, 2024; Yamada & Osaki, 1985). SRGM은 소프트웨어 테스트 과정에서 발견되는 결함이나 코드 수정 과정의 분석 결과를 토대로 향후 결함 발생 가능성을 예측하고 신뢰성 수준을 정량적으로 평가하는 중요한 의사결정 자료를 제공하는 모델이다. 그래서 지금까지 다양한 SRGM 연구가 제안되었으며, 최근 관련된 선행연구는 주로 하나의 최적화 기법(Jin & Jin, 2016; Dhavakumar & Gopalan, 2021)이나 딥러닝(DNN, RNN, LSTM) 모델(Kim, Pham, & Chang, 2023; Kim, Ryu & Baik, 2024)을 사용하여 SRGM 개선을 시도하고 있다. 그러나 이런 접근은 분석상의 한계점이 존재한다. 예를 들어 특정한 하나의 최적화 기법만을 적용하면 국부 최적해(Local Optimum)에 갇힐 위험으로 신뢰성 평가의 정확도 감소를 초래할 수 있다. 그리고 딥러닝 모델을 활용한 추정의 경우에 학습한 데이터 이후에 발생할 수 있는 외삽 문제로 데이터 범위를 벗어난(e.g., 소프트웨어의 테스트 이후에 대한 결함) 예측 검증이 제한되어 소프트웨어 신뢰성을 정확히 판단하기 어렵다.

상기한 제약 문제를 해결하기 위해서 본 연구 목적은 최적화 방법의 최소자승법(Least Squares Method, LSM)과 최대우도법(Maximum Likelihood Estimation, MLE)에 각각 다수의 최적화 알고리즘

을 적용하여 전역 최적해(Global Optimum)에 근접한 값을 찾는 데 있다. 이런 개선된 분석 접근은 최적화 과정에서 국부 최적해에서 벗어나 전체 영역의 탐색을 통해 실제에 근접한 해를 도출하는 데 기여할 수 있을 것이다.

Ⅱ. 소프트웨어 신뢰성 성장 모델(SRGM) 및 최적화 기법

2.1 SRGM(Software Reliability Growth Model)

SRGM은 비동질 포아송 과정(Non-Homogeneous Poisson Process, NHPP)을 기반으로 소프트웨어의 결함 발견 및 수정 과정을 수학적으로 모델링하여 소프트웨어 신뢰성을 정량적으로 평가하고 예측하는 기법이다. NHPP는 시간에 따라 결함 발생률이 변화하는 특성을 반영하며 SRGM은 이를 통해 소프트웨어 테스트 중 결함 발견 패턴을 분석하고 향후 신뢰성 향상 추세를 예측할 수 있다. 다양한 SRGM 모델은 서로 다른 결함 발견 패턴과 신뢰성 향상 추세(e.g. 증가, 감소, S자 커브)를 반영하므로 각 소프트웨어 시스템 및 테스트 환경에 적합한 모델 선택이 중요하다.

본 연구는 여러 소프트웨어 결함 데이터에 대응하여 총 다섯 가지 SRGM(Goel-Okumoto, Delayed S-Shape, Inflection S-Shape, Weibull, Log-Logistic)을 선택한다. 각 모델은 평균값 함수(Mean Value Function, MVF)로 구성되며, 이는 주어진 테스트 동안 누적 결함 수를 계산하는 함수이다. MVF는 소프트웨어 테스트 시간 동안의 결함 발생 패턴을 설명하는 데 사용되며, m(t)로 정의된다. 여기서 t는 소프트웨어 누적 테스트 시간을 나타내고, m(t)는 t 시간까지 발견된 총 결함 수를 의미한다. 또한, 위에 소개한 SRGM 모델들은 공통된 파라미터를 가지고 있으며, 그 공통 파라미터는 데이터의 예측된 총 결함 수인 a와 결함 발생률을 나타내는 b가 있다. 추가적으로 Inflection S-Shape, Weibull, Log-Logistic 모델은 형상 파라미터 c를 포함한다. 이러한 파라미터들은 모델이 소프트웨어 테스트 중 결함 발생 패턴을 어떻게 설명하고 예측하는지 결정짓는 핵심 요소이다.

2.1.1 Goel-Okumoto

Goel-Okumoto 모델은 확률적 모델(Goel & Okumoto, 1979)로 널리 사용되는 SRGM 중 하나이다(IEEE Reliability Society, 2017). 이 모델은 테스트 초기에 많은 결함이 발견되다가 시간이 지나면서 남은 결함의 수가 줄어들어 오류 검출 속도가 점차 느려지는 경향을 설명한다. 즉, 오류가 발견될 확률은 시간이 지남에 따라 감소하며, 이는 소프트웨어 품질이 개선됨에 따라 검출할 수 있는 오류가 점차 줄어드는 현상을 반영한다. Goel-Okumoto 모델의 MVF는 다음과 같다.

$$m(t) = a(1 - e^{-bt}) \tag{1}$$

2.1.2 Delayed S-Shape

Delayed S-Shape 모델(IEEE Reliability Society, 2017; Yamada, Onba, & Osaki, 1983)은 오류 검출에서 소프트웨어 테스트 초기에 오류 발견이 지연되다가 중반부에 빠르게 증가하고 후반부로 갈수록 다시 감소하는 S자 형태의 패턴을 설명한다. 테스트 초기에는 오류를 발견하는 데 시간이 소요되지만, 테스트 진행에서 오류 발견 수가 급격히 증가하고, 이후 남은 오류가 줄어들면서 검출 속도가 다시 감소하는 구조이다. Delayed S-Shape 모델의 MVF는 다음과 같다.

$$m(t) = a(1-(1+bt)e^{-bt}) \tag{2}$$

2.1.3 Inflection S-Shape

Inflection S-Shape 모델(Ohba, 1984)은 초반 단계에서 오류 발견 속도가 느리다가 중반에 급격히 증가하고 이후 다시 감소하는 S자 형태의 그래프를 형성한다. 해당 모델은 초기 단계에서 오류 검출이 지연되는 Delayed S-Shape 모델과 달리 상대적으로 더 빠르고 활발한 오류 검출 패턴을 보인다. Inflection S-Shape 모델의 MVF는 다음과 같다.

$$m(t) = a\left(\frac{1-e^{-bt}}{1+ce^{-bt}}\right) \tag{3}$$

2.1.4 Weibull

Weibull 모델(IEEE Reliability Society, 2017; Kenney, 1993)은 형상 파라미터를 사용하여 오류 발생률의 변화를 설명하므로 고장 시간이 비정상적이거나 다양한 분포를 따르는 시스템의 신뢰성 평가에 유용하다(e.g., Verma, Anand, Kapur, & Aggarwal, 2022). 형상 파라미터는 오류 발생 패턴을 결정하며, 그 값이 1보다 작을 경우 오류 발생률은 시간이 지남에 따라 감소하고, 1일 경우 일정하며 1보다 클 경우에 오류 발생률의 증가 경향이 나타난다. Weibull 모델의 MVF는 다음과 같다.

$$m(t) = a(1-e^{-bt^c}) \tag{4}$$

2.1.5 Log-Logistic

Log-Logistic 모델(IEEE Reliability Society, 2017; Gokhale & Trivedi, 1998)은 시간이 지남에 따라 처음 오류 검출 비율이 증가하다가 일정 시점 이후에 감소하는 S자 패턴 모델이다. 이 모델은 Delayed S-Shape 모델보다 초기 오류 검출이 빠르며, Inflection S-Shape 모델과 비교하여 오류 검출의 증가와 감소가 더 완만한 패턴을 따른다. Log-Logistic 모델의 MVF는 다음과 같다.

$$m(t) = a\left(\frac{(bt)^c}{1+(bt)^c}\right) \tag{5}$$

2.2 최적화 기법

2.2.1 LSM

LSM은 주어진 데이터에 대한 모델의 파라미터 추정에 사용되는 통계적 기법이다. 이 방법은 관측된 데이터와 모델이 예측하는 값 간의 차이를 최소화하는 방향으로 파라미터를 조정하여 모델의 적합도를 극대화한다. 오차 제곱합(Sum of Squared Error, SSE)을 최소화하는 파라미터를 찾고자 수학적으로 SSE는 다음과 같은 목적 함수로 정의한다.

$$SSE = \sum_{i=1}^{n}(y_i - f(x_i;\theta))^2 \tag{6}$$

여기서 y_i는 관측된 값, $f(x_i;\theta)$는 모델의 예측값, θ는 추정해야 할 파라미터 벡터를 나타낸다. 최소자승법은 이 SSE를 최소화하는 파라미터 θ를 찾음으로써 모델이 관측된 데이터를 잘 설명할 수 있도록 한다.

2.2.2 MLE

MLE는 주어진 데이터가 관찰될 확률을 최대화하는 파라미터 값을 찾는 통계적 기법이다. 이 방법은 관측 데이터와 모델 간의 적합도를 극대화하여 데이터가 주어졌을 때 가장 가능성이 높은 파라미터 값을 추정할 수 있다. 최대우도법의 기본 개념은 다음과 같다. 관측된 데이터 $\{x_1, x_2, \ldots, x_n\}$가 특정 확률 분포 $f(x_i;\theta)$를 따르고, 여기서 θ는 추정해야 할 파라미터 벡터라고 가정할 때, MLE는 이 확률 분포에서 관측된 데이터가 나타날 확률, 즉 우도(Likelihood)를 최대화하는 파라미터 θ를 찾는 것이다. 우도 함수(Likelihood Function)는 다음과 같이 정의된다.

$$L(\theta) = \prod_{i=1}^{n} f(x_i;\theta) \tag{7}$$

이 함수는 관측된 각 데이터 포인트가 주어진 파라미터 θ에서 발생할 확률의 곱으로 표현된다. MLE는 이 우도 함수 계산을 간소하기 위해 로그를 취해 로그-우도 함수(Log-Likelihood Function)로 변환하고, 이를 최대화하는 파라미터를 찾는다. 해당 함수는 다음과 같다.

$$l(\theta) = \sum_{i=1}^{n} \log f(x_i;\theta) \tag{8}$$

Ⅲ. 다양한 알고리즘을 적용한 SRGM 최적화 기법

SRGM의 최적화 성능 향상 방법으로 다음 세 가지를 제시한다. 첫째, LSM과 MLE에 적합한 여러 최적화 알고리즘을 적용하여 국부 최적해가 아닌 전역 최적해에 수렴하는 최적화 방법을 도입한다. 둘째, 방산 분야와 유사한 소프트웨어 신뢰성 사례로 NASA 프로젝트의 소프트웨어 결함 데이터[1]를 사용한다. 셋째, 성능이 가장 우수한 SRGM, 최적화 방법, 최적화 알고리즘의 조합을 선별하기 위해 회귀 분석에서 사용되는 대표적인 지표를 활용한다.

3.1 최적화 알고리즘

본 연구는 SciPy 라이브러리(Virtanen et al., 2020)[2]를 사용하여 SRGM 최적화를 진행한다. LSM 최적화는 SciPy의 Optimize 모듈에서 제공하는 ‘least_squares’ 함수를 사용하며, MLE 최적화 방식은 ‘minimize’ 함수를 이용한다. ‘minimize’ 함수는 목표 함수의 최솟값을 찾는 함수이므로, 이를 MLE 최적화에 적용할 때는 로그-우도 함수에 반대 부호를 취해 최대화 문제를 최소화 문제로 변환하여 해결한다. ‘least_squares’ 함수와 ‘minimize’ 함수는 각각 머신러닝에서 사용되는 여러 최적화 알고리즘을 제공하며, ‘least_squares’ 함수는 총 3개 최적화 알고리즘, ‘minimize’ 함수는 총 15개의 최적화 알고리즘을 포함한다(Table 1).

〈Table 1〉 Type of solver in ‘least_squares’ and ‘minimize’ function

Algorithms	Bounds option	Algorithms	Bounds option
trf	O	TNC	O
dogbox	O	COBYLA	O
lm	X	COBYQA	O
Nelder-Mead	O	SLSQP	O
Powell	O	trust-constr	O
CG	X	dogleg	X
BFGS	X	trust-ncg	X
Newton-CG	X	trust-krylov	X
L-BFGS-B	O	trust-exact	X

상기한 LSM 및 MLE 최적화 방식을 합쳐 총 18개 중, 실제 적용되는 최적화는 파라미터 경계값 지정을 지원하는 알고리즘을 선택한다. SRGM의 각 파라미터는 역할에 따라 Table 2와 같이 경곗값

1) Lyu, M. R. (Ed.). (1996). *Handbook of software reliability engineering*. McGraw-Hill.

2) SciPy는 파이썬 기반의 과학 및 공학 계산을 위한 오픈 소스 라이브러리로 여러 수치 최적화 알고리즘을 포함한다.

을 지정할 수 있다. a는 총 예측 결함 수로 발견된 총 예측 결함 수보다 크거나 같아야 하며, b는 결함율을 나타내므로 0보다 커야 하고, c는 형상 파라미터로 0보다 큰 값으로 정의한다. 파라미터 경계값을 지정하면, 최적화 과정에서 각 파라미터가 주어진 범위 내에서만 최적화되도록 제한을 두게 되어 계산 결과의 안정성과 정확성을 높일 수 있다.

〈Table 2〉 Range of parameters in SRGMs

Parameters	Range
a	a ≥ Total number of failures found
b	b 〉 0
c	c 〉 0

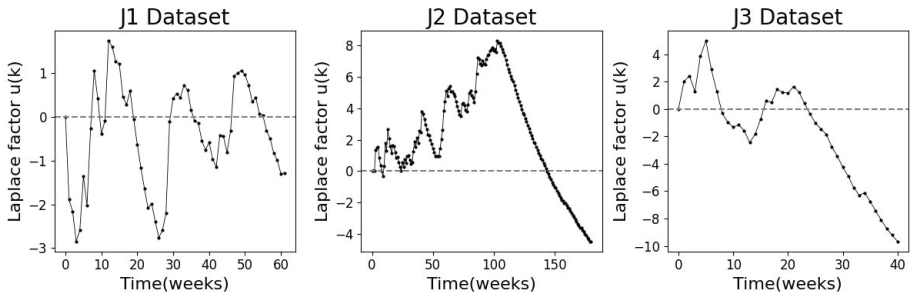
따라서 Table 1과 같이 LSM 최적화 방식은 2개의 최적화 알고리즘, MLE 최적화 방식에서는 COBYLA 알고리즘이 최적화 과정에서 원활하게 수행되지 않아 이를 제외하고 총 7개의 최적화 알고리즘을 사용한다. 이를 통해 동일한 소프트웨어 결함 데이터에서 SRGM 당 총 9가지의 결과를 비교할 수 있으며, 이를 바탕으로 전역 최적해에 가까운 결과값을 도출할 수 있다.

3.1.1 최적화 알고리즘에 적용한 데이터

본 연구에 적용할 데이터는 방산 분야와 유사한 소프트웨어 결함 데이터를 사용하기 위해 M. R. Lyu의 “Handbook of Software Reliability Engineering”에 실린 NASA의 JPL(Jet Propulsion Laboratory)에서 진행한 프로젝트 데이터를 활용한다. 총 3개의 데이터셋이 사용되며, 각 데이터셋의 명칭은 J1, J2, J3이다.

- J1은 주석이 없는 약 14,000줄의 코드로 구성되어 있으며, KLOC(Kilo Lines of Code)당 약 9.5개의 결함이 발견되었다.
- J2는 주석이 없는 약 7,000줄의 코드로 구성되어 있으며, KLOC당 약 10.2개의 결함이 발견되었다.
- J3는 J2의 테스트 프로파일의 일부 요소를 재구성한 것으로, KLOC당 약 10.1개의 결함이 발견되었다.

Figure 1은 J1, J2, J3 데이터에 대해 라플라스 테스트(Kanoun & Laprie, 1994)를 적용한 그래프이다. u(k)는 라플라스 인자로 식 (9)와 같이 계산되며, u(k)가 양수일 때는 소프트웨어 결함이 증가하는 추세를 나타내고, 음수일 때는 감소하는 추세를 나타낸다.



〈Figure 1〉 Laplace trend plots of J1, J2 and J3

J1은 소프트웨어 결함의 증가-감소가 반복되며 점차 늘어나는 경향이며, J2와 J3는 소프트웨어 결함이 증가하다가 점차 감소하는 경향으로 나타난다. J2는 J3에 비해 중간에 소프트웨어 결함이 가파르게 증가하는 그래프이다. 이처럼 테스트 환경 및 소프트웨어 시스템마다 결함 추세가 다양하므로 소프트웨어 신뢰성 분야는 여러 종류의 SRGM을 적용한다.

$$u(k) = -\frac{\sum_{i=1}^{k} N(i) - \frac{k+1}{2} N(k)}{\sqrt{\frac{k^2-1}{12} N(k)}} \tag{9}$$

3.1.2 최적화 알고리즘에 적용한 성능 지표

SRGM 최적화 결과의 성능을 비교하기 위해 본 연구에서는 회귀 분석에서 널리 사용되는 여러 성능 지표를 활용하였다. 사용된 지표는 평균 제곱 오차(Mean Squared Error, MSE), 평균 제곱근 오차(Root Mean Squared Error, RMSE), 평균 절대 오차(Mean Absolute Error, MAE), 평균 절대 백분율 오차(Mean Absolute Percentage Error, MAPE), 결정 계수(R^2 Score)이다. 그러나 위 지표는 모두 예측값과 실제값 간의 차이를 기반으로 계산하며, 각 예측 결과 간의 성능 차이가 유사하게 나타나는 경향을 보였다. 또한, 소프트웨어 결함 데이터의 특성상 이상치에 대한 민감한 비교가 필요하지 않다는 점을 고려하여 SRGM 최적화 결과 분석과 비교는 MSE 성능 평가 지표를 활용한다. 이 지표는 예측값과 실제값 간의 차이를 제곱한 후 그 값을 평균하여 계산한 가장 대표적 지표이다.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{10}$$

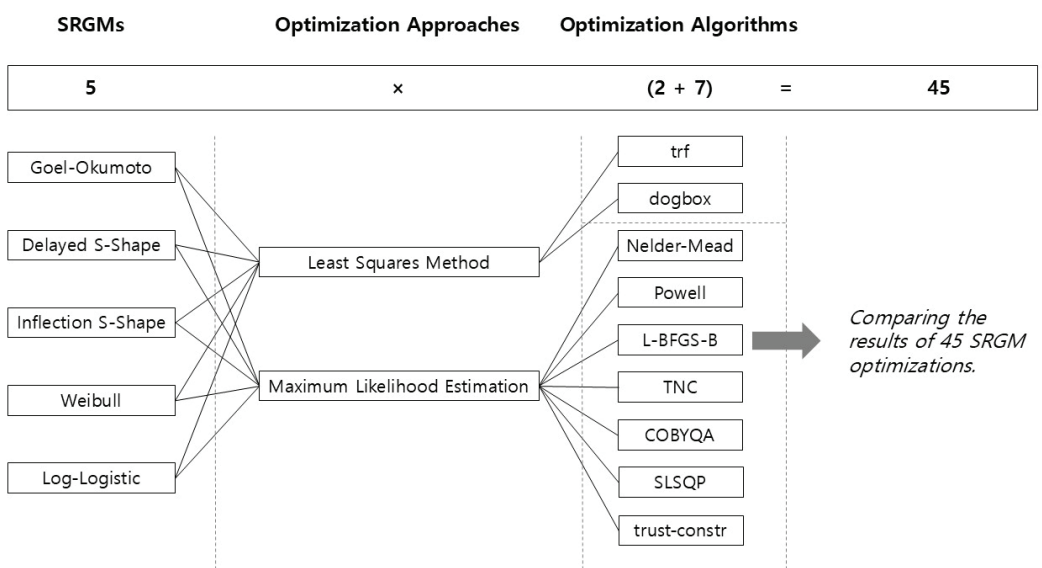
모델의 설명력을 판단하기 위한 지표는 R^2 score만을 보고한다. 해당 결정계수는 실제값의 전체 분산 중 예측 모델이 설명하는 분산 비율을 나타내며 값이 1에 가까울수록 모든 데이터가 모델을

완벽하게 설명한다는 것으로 모델의 설명력이 좋다고 해석할 수 있다. 반면, 값이 0에 가까우면 설명력이 낮고 음수일 경우에는 모델의 성능이 단순 평균보다도 떨어진다는 것이다.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{11}$$

Ⅳ. SRGM 최적화 결과 분석 및 비교

앞서 설명한 SRGM과 최적화 방식 및 최적화 알고리즘을 조합하여 총 45개를 도출하였다(Figure 2). 즉, 5종의 SRGM, LSM 방식에 2종, MLE 방식에는 7종의 최적화 알고리즘을 적용하고, 각 조합에 대해 MSE와 R^2 Score를 산출하고 이를 분석한다.



〈Figure 2〉 Optimization process of SRGMs (Software Reliability Growth Models)

4.1 J1 최적화 결과 분석 및 비교

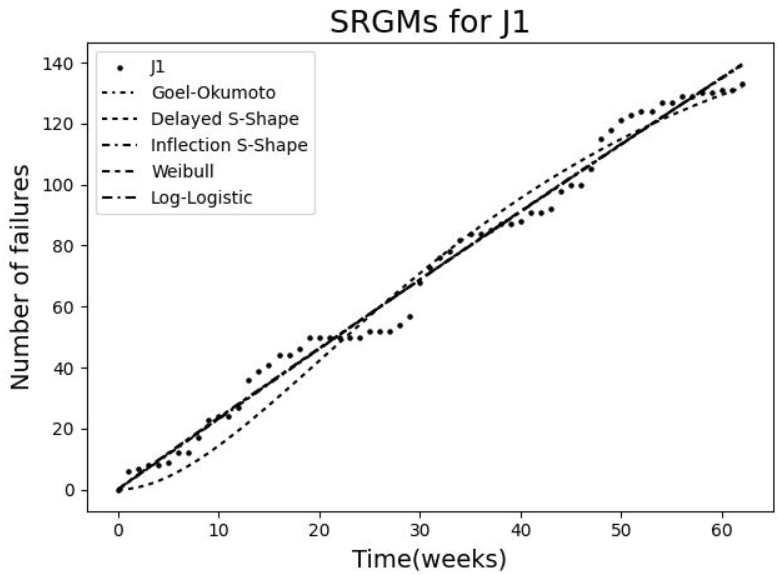
J1에 관한 45개의 MSE 결과와 설명량을 도출한 결과(Table 3), MSE가 가장 낮은 최적화 결과는 20.838로 Weibull 모델 LSM 최적화 방식의 trf 최적화 알고리즘이다. 반면, MSE가 가장 높은 최적화

결과는 156.796(약 7.5배 차이)로 Weibull 모델 LSM 최적화 방식의 dogbox 최적화 알고리즘이다. 이는 Weibull 모델 LSM 최적화 방식의 trf 최적화 알고리즘이 전역 최적해에 근접하지만, 동일한 SRGM과 최적화 방식에 대해 dogbox 최적화 알고리즘은 국부 최적해에 갇힌 것을 보인다. Weibull 모델 외에도 Goel-Okumoto 모델은 MLE 최적화 방식에서 Nelder-Mead 알고리즘이 MSE 41.28로 MSE가 최소(20.892)인 결과에 비해 약 2배 차이로 국부 최적해에 머무른 것으로 확인된다. R^2 score도 대부분의 결과가 0.97-0.99 범위로 총 44개의 예측 방식이 실제 데이터에 대해 높은 수준의 설명력을 보여준다. 그러나 앞서 국부 최적해에 머물렀다고 판단한 Weibull 모델 MLE 최적화 방식의 dogbox 최적화 알고리즘은 0.9로 상대적으로 낮은 것을 나타낸다.

〈Table 3〉 Result summary table with 45 optimization for J1

SRGMs	Optimization approaches	Optimization algorithms	MSE	R^2 score	SRGMs	Optimization approaches	Optimization algorithms	MSE	R^2 score
Goel-Okumoto	LSM	trf	20.892	0.987	Inflection S-Shape	MLE	TNC	26.247	0.984
		dogbox	20.892	0.987			COBYQA	26.247	0.984
	MLE	Nelder-Mead	41.28	0.975			SLSQP	26.247	0.984
		Powell	27.071	0.983			trust-constr	26.247	0.984
		L-BFGS-B	27.071	0.983	Weibull	LSM	trf	20.838	0.987
		TNC	27.071	0.983			dogbox	156.796	0.906
		COBYQA	27.071	0.983		MLE	Nelder-Mead	28.786	0.982
		SLSQP	27.071	0.983			Powell	27.695	0.983
		trust-constr	27.072	0.983			L-BFGS-B	28.094	0.983
Delayed S-Shape	LSM	trf	47.757	0.971			TNC	28.094	0.983
		dogbox	47.757	0.971			COBYQA	28.093	0.983
	MLE	Nelder-Mead	51.406	0.969			SLSQP	28.093	0.983
		Powell	51.317	0.969			trust-constr	28.093	0.983
		L-BFGS-B	51.341	0.969	Log-Logistic	LSM	trf	20.84	0.987
		TNC	51.341	0.969			dogbox	22.603	0.986
		COBYQA	51.341	0.969		MLE	Nelder-Mead	28.209	0.983
		SLSQP	51.344	0.969			Powell	27.149	0.983
		trust-constr	51.341	0.969			L-BFGS-B	28.209	0.983
Inflection S-Shape	LSM	trf	20.872	0.987			TNC	28.21	0.983
		dogbox	20.872	0.987			COBYQA	28.21	0.983
	MLE	Nelder-Mead	26.247	0.984			SLSQP	28.209	0.983
		Powell	26.258	0.984			trust-constr	28.209	0.983
		L-BFGS-B	26.247	0.984					

SRGM별 전역 최적해에 근사한 결과를 비교해 보면, Goel-Okumoto, Inflection S-Shape, Weibull, Log-Logistic 모델은 각각 MSE가 20.892, 20.872, 20.838, 20.840으로 유사한 수준을 보인다. 반면, Delayed S-Shape 모델은 MSE가 47.757로 앞선 네 모델에 비해 약 두 배 이상 높은 값을 나타낸다. 전역 최적해 근사 결과의 시각화(Figure 3)[3]를 보면, MSE가 유사한 네 가지 SRGM 모델은 소프트웨어 결함의 증가 추세로 J1 데이터와 유사한 패턴을 따르는 것으로 나타났다. 반면, 상대적으로 MSE가 높은 Delayed S-Shape 모델은 실제 데이터와 비교하여 지나치게 S자 곡선의 경향을 보인다.



〈Figure 3〉 Graph of SRGMs with minimum MSE for J1

4.2 J2 최적화 결과 분석 및 비교

Table 4는 J2의 경우에 45개 MSE 결과와 R^2 score 결과이다. MSE가 가장 낮은 값은 60.576으로 Inflection S-Shape 모델 LSM 최적화 방식의 trf 최적화 알고리즘이다. 이는 전역 최적해에 가까운 해가 도출되어 데이터에 제일 적합한 SRGM임을 의미한다. 반면, 다음 조합은 상대적으로 높은 MSE 값을 보인다. Goel-Okumoto 모델의 MLE 방식과 Nelder-Mead 알고리즘(MSE 1278.685), Delayed S-Shape 모델의 MLE 방식과 SLSQP 알고리즘(MSE 4820.465), Inflection S-Shape 모델의 LSM 방식과 dogbox 알고리즘(MSE 850.026), Weibull 모델의 MLE 방식과 Nelder-Mead(MSE 1243.349) 및 Powell(MSE 15767.680) 알고리즘, Log-Logistic 모델의 MLE 방식과 Nelder-Mead(MSE 1243.349) 및 Powell(MSE 11005.344) 알고리즘이다.

3) 시각화에 필요한 각 SRGM의 추정 파라미터 수치는 <부록 1>, <부록 2>, <부록 3>, <부록 4>, <부록 5>로 기술됨.

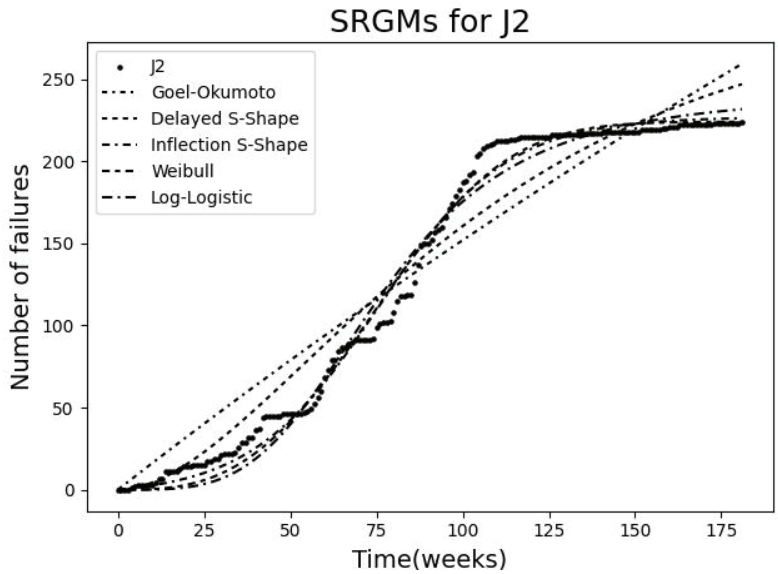
이 결과는 각 모델의 최소 MSE 값 613.917, 311.451, 60.576, 70.778, 121.754와 비교할 때, 최소 약 2배에서 최대 약 222배까지 차이가 나타났다. 특히 Delayed S-Shape 모델 MLE 최적화 방식의 SLSQP 최적화 알고리즘은 R^2 score 값이 0.35로 예측 모델이 데이터를 잘 설명하지 못하고 있으며, Weibull 모델과 Log-Logistic 모델의 MLE 최적화 방식에서 Powell 최적화 알고리즘은 각각 –1.123, -0.482로 데이터에 대해 모델 예측이 좋지 않음을 보여준다.

〈Table 4〉 Result summary table with 45 optimization for J2

SRGMs	Optimization approaches	Optimization algorithms	MSE	R^2 score	SRGMs	Optimization approaches	Optimization algorithms	MSE	R^2 score
Goel-Okumoto	LSM	trf	613.917	0.917	Inflection S-Shape	MLE	TNC	63.535	0.991
		dogbox	613.917	0.917			COBYQA	62.61	0.991
	MLE	Nelder-Mead	1278.685	0.827			SLSQP	62.612	0.991
		Powell	822.292	0.889			trust-constr	62.611	0.991
		L-BFGS-B	822.292	0.889	Weibull	LSM	trf	70.778	0.99
		TNC	822.292	0.889			dogbox	275.782	0.962
		COBYQA	822.292	0.889		MLE	Nelder-Mead	1243.349	0.832
		SLSQP	822.292	0.889			Powell	15767.68	-1.123
		trust-constr	822.295	0.889			L-BFGS-B	102.362	0.986
Delayed S-Shape	LSM	trf	311.451	0.958			TNC	447.716	0.939
		dogbox	311.451	0.958			COBYQA	283.203	0.961
	MLE	Nelder-Mead	417.674	0.943			SLSQP	102.028	0.986
		Powell	417.617	0.943			trust-constr	101.353	0.986
		L-BFGS-B	417.617	0.943	Log-Logistic	LSM	trf	121.754	0.983
		TNC	417.617	0.943			dogbox	121.754	0.983
		COBYQA	417.617	0.943		MLE	Nelder-Mead	1243.349	0.832
		SLSQP	4820.465	0.35			Powell	11005.344	-0.482
		trust-constr	417.618	0.943			L-BFGS-B	199.929	0.973
Inflection S-Shape	LSM	trf	60.576	0.991			TNC	199.929	0.973
		dogbox	850.026	0.885			COBYQA	199.929	0.973
	MLE	Nelder-Mead	62.61	0.991			SLSQP	199.938	0.973
		Powell	62.612	0.991			trust-constr	199.929	0.973
		L-BFGS-B	62.611	0.991					

SRGM별로 전역 최적해에 근사한 결과를 비교해 보면, Inflection S-Shape와 Weibull 모델의 MSE 값은 60.576, 70.778로 유사하다. 반면, Goel-Okumoto, Delayed S-Shape, Log-Logistic 모델은 각각 MSE 613.917, 311.451, 121.754로 적게는 앞의 모델에 비해 약 2배에서 높게는 약 10배까지의 차이를 보인다. 이러한 MSE 결과를 토대로 Figure 4를 보면, Inflection S-Shape와 Weibull 모델은 S자 형

태의 경향으로 J2 데이터를 잘 나타내지만, Goel-Okumoto, Delayed S-Shape, Log-Logistic 모델은 상대적 증가 추세이다.



〈Figure 4〉 Graph of SRGMs with minimum MSE for J2

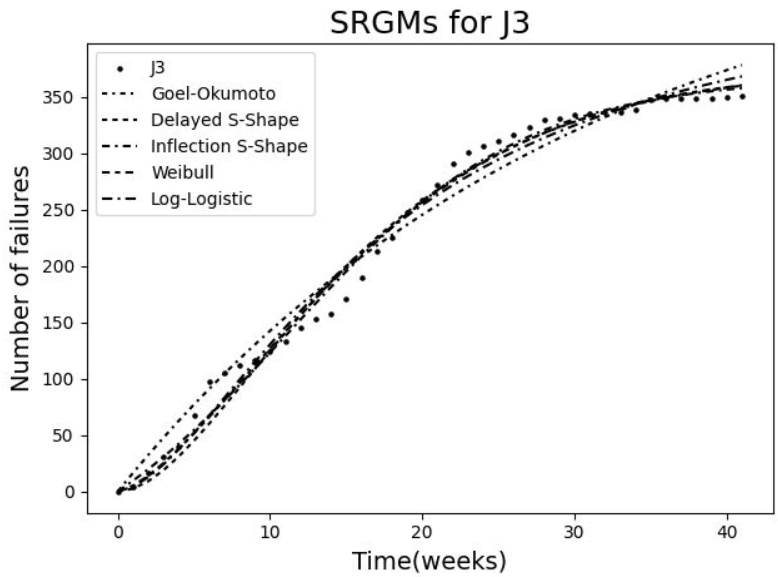
4.3 J3 최적화 결과 분석 및 비교

Table 5는 J3에 대해 45개의 MSE 결과와 R^2 score를 도출한 값이다. MSE가 가장 낮은 값은 122.261로 Inflection S-Shape 모델 LSM 최적화 방식의 trf, dogbox 최적화 알고리즘이다. 나머지 결과를 비교하면, Goel-Okumoto, Delayed S-Shape, Inflection S-Shape, Weibull, Log-Logistic 모델 순서대로 MSE가 각각 321.202～477.027, 190.616～227.633, 122.261～132.347, 155.694～183.44, 203.982～278.549이다. 이는 최대 차이가 약 1.4배 이하로 모든 최적화 방식과 알고리즘 조합이 전역 최적해에 근접했다고 본다. 또한, R^2 score의 모든 결과도 0.965 이상으로 제시된 최적화 방식과 최적화 알고리즘의 조합이 모두 전역 최적해에 근접한 값을 추정하였다고 볼 수 있다.

〈Table 5〉 Result summary table with 45 optimization for J3

SRGMs	Optimization approaches	Optimization algorithms	MSE	R^2 score	SRGMs	Optimization approaches	Optimization algorithms	MSE	R^2 score
Goel-Okumoto	LSM	trf	321.202	0.976	Inflection S-Shape	MLE	TNC	132.347	0.99
		dogbox	321.202	0.976			COBYQA	132.347	0.99
	MLE	Nelder-Mead	477.027	0.965			SLSQP	132.347	0.99
		Powell	476.883	0.965			trust-constr	132.347	0.99
		L-BFGS-B	476.883	0.965	Weibull	LSM	trf	155.694	0.988
		TNC	476.883	0.965			dogbox	155.694	0.988
		COBYQA	476.883	0.965		MLE	Nelder-Mead	183.44	0.986
		SLSQP	476.884	0.965			Powell	182.795	0.986
		trust-constr	476.884	0.965			L-BFGS-B	183.434	0.986
Delayed S-Shape	LSM	trf	190.616	0.986			TNC	183.433	0.986
		dogbox	190.616	0.986			COBYQA	183.435	0.986
	MLE	Nelder-Mead	227.633	0.983			SLSQP	183.425	0.986
		Powell	227.435	0.983			trust-constr	183.434	0.986
		L-BFGS-B	227.435	0.983	Log-Logistic	LSM	trf	203.982	0.985
		TNC	227.435	0.983			dogbox	203.982	0.985
		COBYQA	227.435	0.983		MLE	Nelder-Mead	277.056	0.979
		SLSQP	227.435	0.983			Powell	278.549	0.979
		trust-constr	227.435	0.983			L-BFGS-B	277.064	0.979
Inflection S-Shape	LSM	trf	122.261	0.991			TNC	277.063	0.979
		dogbox	122.261	0.991			COBYQA	277.064	0.979
	MLE	Nelder-Mead	132.346	0.99			SLSQP	277.065	0.979
		Powell	132.347	0.99			trust-constr	277.064	0.979
		L-BFGS-B	132.347	0.99					

J3 데이터에 대한 SRGM별 MSE 최솟값을 살펴보면, Goel-Okumoto, Delayed S-Shape, Inflection S-Shape, Weibull, Log-Logistic 모델 순으로 각각 321.202, 190.616, 122.261, 155.694, 203.982 값이 나타났다. 전반적으로 MSE는 유사한 수준을 보이지만, Goel-Okumoto 모델은 최솟값 대비 약 2.6배 높은 수치로 상대적 성능이 떨어진다. Figure 5를 보면, 대부분의 모델은 S자 형태를 띠는 J3 데이터 특성이 잘 반영되었으나 Goel-Okumoto 모델은 초반에 급격히 증가하는 결함 패턴을 적절히 추정하지 못한 경향을 보였다.



〈Figure 5〉 Graph of SRGMs with minimum MSE for J3

Ⅴ. 결론 및 논의

본 연구는 소프트웨어 신뢰성 평가에서 SRGM에 단일 최적화 방식의 위험성을 입증하였다. J1, J2, J3 데이터에 대해 최적화 알고리즘에 따른 MSE 차이가 최대 약 222배(수치적으로 15696.9 차이)(Table 4)에 해당하는 결과를 보여주었으며, 이를 통해 여러 최적화 알고리즘을 적용하여 전역 최적해에 가장 근접한 결과를 도출하는 것이 중요함을 강조하였다. 본 연구에서 활용한 세 개의 데이터셋은 공통적으로 LSM 방식의 trf 최적화 알고리즘이 낮은 MSE 값을 도출하였다. 하지만, LSM 방식의 dogbox와 MLE 방식의 Nelder-Mead, Powell, SLSQP 최적화 알고리즘은 상대적으로 큰 MSE 값을 보이며 전역 최적해의 결과와 유의미한 차이를 드러냈다. 이는 SRGM에 적합한 최적화 방식과 알고리즘의 조합들을 식별할 수 있거나 국부 최적해에 머무를 수 있는 비적합한 조합을 통계적으로 검증할 수 있음을 시사한다.

상기한 연구결과를 토대로 본 연구는 SRGM에 적합한 최적화 방식과 알고리즘의 조합 식별이나 비적합한 조합 제거를 통해 최적화 목적 계산에서 자원 소모를 최소화하면서 분석소요 시간도 단축할 수 있어 실무적 적용 측면에서 의의가 있다. 특히, 국방 및 방산 분야의 소프트웨어에 적합한 소프트웨어 신뢰성 평가에서 국방 무기체계에 특화된 결함 데이터를 활용한 분석이 필요하다. 이런 점에서 본 연구는 SRGM 최적화를 통해 무기체계 소프트웨어의 신뢰성 평가에 관한 정확성 향상 방안을 제시하였다는 점에서 연구가 시사하는 바가 크다.

Acknowledgements

이 논문은 2023년도 대한민국 정부(산업통상자원부, 방위사업청)의 재원으로 국방과학연구소 민군협력진흥원에서 수행하는 민군기술이전사업의 연구비 지원을 받아 진행된 연구이다(No. 23-SF-AI-05, AI 기반 소프트웨어 신뢰도 분석 및 관리 프로그램 개발, 2023.11～2025.10).

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Author contributions

Conceptualization: SM and PS, Literature review: SM and PS, Resources and Data curation: SM, JJ, and LJ, Investigation and Methodology: SM and PS, Writing (Original Draft): SM, Project administration and Supervision: RI and PS.

Reference

Bahnam, B. S., Dawwod, S. A., & Younis, M. C. (2024). Optimizing software reliability growth models through simulated annealing algorithm: parameters estimation and performance analysis. *The Journal of Supercomputing, 80*(11), 16173-16201. https://doi.org/10.1007/s11227-024-06046-4

Chatterjee, S., Saha, D., & Sharma, A. (2021). Multi-upgradation software reliability growth model with dependency of faults under change point and imperfect debugging. *Journal of Software: Evolution and Process, 33*(6), e2344. https://doi.org/10.1002/smr.2344

Cho, Y., Min, S., Lim, S., & Choi, K. (2023). A study on the differences in the perceived importance of jet fighter performance improvement factors. *Journal of Advances in Military Studies, 6*(2), 17-41. https://doi.org/10.37944/jams.v6i2.192

Dhavakumar, P., & Gopalan, N. P. (2021). An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing, 12*, 3177-3188. https://doi.org/10.1007/s12652-020-02476-z

Goel, A. L., & Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, R-28(3), 206-211. https://doi.org/10.1109/TR.1979.5220566

Gokhale, S. S., & Trivedi, K. S. (1998). Log-logistic software reliability growth model. In *Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium* (pp. 34-41). IEEE.

Hur, Y. (2024). Exploring the critical factors of depot maintenance on weapon systems in South Korea. *Journal of Advances in Military Studies, 7*(3), 1-12. https://doi.org/10.37944/jams.v7i3.271

IEEE Reliability Society. (2017). *IEEE Std. 1633-2016 - IEEE recommended practice on software reliability*. IEEE. https://doi.org/10.1109/IEEESTD.2017.7827907

Jin, C., & Jin, S. (2016). Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization. *Applied Soft Computing*, *40*, 283-291. https://doi.org/10.1016/j.asoc.2015.11.041

Kanoun, K., & Laprie, J. C. (1994). Software reliability trend analyses from theoretical to practical considerations. *IEEE Transactions on Software Engineering*, *20*(9), 740-747. https://doi.org/10.1109/32.317434

Kenney, G. Q. (1993). Estimating defects in commercial software during operational use. *IEEE Transactions on Reliability*, *42*(1), 107-115. https://doi.org/10.1109/24.210280

Kim, Y. S., Pham, H., & Chang, I. H. (2023). Deep-learning software reliability model using SRGM as activation function. *Applied Sciences, 13*(19), 10836. https://doi.org/10.3390/app131910836

Kim, T., Ryu, D., & Baik, J. (2024). Automated Machine Learning for Enhanced Software Reliability Growth Modeling: A Comparative Analysis with Traditional SRGMs. *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)* (pp. 483-493) https://doi.org/10.1109/QRS62785.2024.00055

Kim, Y., Kim, D., & Jeong, D. (2024). Feasibility and foundational elements of CBM+ technology application in weapon systems. *Journal of Advances in Military Studies, 7*(2), 29-54. https://doi.org/10.37944/jams.v7i2.243

Oh, J. W., Eom, W. Y., Chae, S. Y., Cho, H. B., & Chang, J. H. (2024). A Study of Feedback Method for Weapon System Software Defection Analysis based on Software Risk Analysis. *Journal of Korea Academia-Industrial cooperation Society, 25*(6), 371-377. https://doi.org/10.5762/KAIS.2024.25.6.371

Ohba, M. (1984). Inflection S-shaped software reliability growth model. In S. Osaki & Y. Hatoyama (Eds.), *Stochastic Models in Reliability Theory* (Vol. 235, pp. 144-162). Springer.

Samal, U. (2024). Software Reliability Growth Model Considering Imperfect Debugging and Fault Removal Efficiency. *Quality and Reliability Engineering International, 41*(4), 1268-1278. https://doi.org/10.1002/qre.3716

Samal, U., & Kumar, A. (2024). Metrics and trends: a bibliometric approach to software reliability growth models. *Total Quality Management & Business Excellence, 35*(11-12), 1274-1295. https://doi.org/10.1080/14783363.2024.2366510

Sung, S. I., Yoon, H. J., & Lim. J. H. (2021). Considerations on Laws and Regulations for Improving the Reliability of Weapons Systems: Based on the Reliability Growth Model. *Journal of Applied Reliability, 21*(2), 181-189. https://doi.org/10.33162/JAR.2021.6.21.2.181

Virtanen, P., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, J., & Zhang, C. (2025). Optimization Selection Method for Software Reliability Growth Model Based on Cosine Similarity. *Journal of Systems and Software, 228*, 112474. https://doi.org/10.1016/j.jss.2025.112474

Yamada, S., & Osaki, S. (1985). Software reliability growth modeling: Models and applications. *IEEE Transactions on Software Engineering*, SE-11(12), 1431-1437. https://doi.org/10.1109/TSE.1985.232179

Yamada, S., Onba, M., & Osaki, S. (1983). S-shaped reliability growth modeling for software error detection. *IEEE Transactions on Reliability*, R-32(5), 475-484. https://doi.org/10.1109/TR.1983.5221735

원 고 접 수 일 2025년 03월 06일
원 고 수 정 일 2025년 04월 18일
게 재 확 정 일 2025년 05월 13일

부록(Appendix)

〈부록 1〉 Goel-Okumoto parameters for the J1, J2 and J3 datasets

Optimization approaches	Optimization algorithms	J1		J2		J3	
		a	b	a	b	a	b
LSM	trf	1781.28	1.31e-3	1089.84	1.5e-3	514.32	3.24e-2
	dogbox	1781.28	1.31e-3	1089.85	1.5e-3	514.32	3.24e-2
MLE	Nelder-Mead	143959403.79	1.49e-8	83051965.79	1.49e-8	413.23	4.62e-2
	Powell	414.06	6.25e-3	344.4	5.81e-3	413.3	4.61e-2
	L-BFGS-B	414.06	6.25e-3	344.4	5.81e-3	413.3	4.61e-2
	TNC	414.06	6.25e-3	344.4	5.81e-3	413.3	4.61e-2
	COBYQA	414.06	6.25e-3	344.4	5.81e-3	413.3	4.61e-2
	SLSQP	414.07	6.25e-3	344.4	5.81e-3	413.3	4.61e-2
	trust-constr	414.03	6.25e-3	344.39	5.81e-3	413.3	4.61e-2

〈부록 2〉 Delayed S-Shape parameters for the J1, J2 and J3 datasets

Optimization approaches	Optimization algorithms	J1		J2		J3	
		a	b	a	b	a	b
LSM	trf	162.93	4.93e-2	291.9	1.84e-2	377.37	1.18e-1
	dogbox	162.93	4.93e-2	291.9	1.84e-2	377.37	1.18e-1
MLE	Nelder-Mead	160.55	5.16e-2	244.79	2.26e-2	362.6	1.29e-1
	Powell	160.61	5.15e-2	244.81	2.26e-2	362.62	1.29e-1
	L-BFGS-B	160.6	5.15e-2	244.81	2.26e-2	362.62	1.29e-1
	TNC	160.6	5.15e-2	244.81	2.26e-2	362.62	1.29e-1
	COBYQA	160.6	5.15e-2	244.81	2.26e-2	362.62	1.29e-1
	SLSQP	160.59	5.15e-2	61585807303256.37	1.49e-8	362.62	1.29e-1
	trust-constr	160.6	5.15e-2	244.81	2.26e-2	362.62	1.29e-1

〈부록 3〉 Inflection S-Shape parameters for the J1, J2 and J3 datasets

Optimization approaches	Optimization algorithms	J1			J2			J3		
		a	b	c	a	b	c	a	b	c
LSM	trf	435.26	1.09e-2	1.06	226.93	5.4e-2	59.62	367.52	1.28e-1	4.16
	dogbox	435.23	1.09e-2	1.06	224	1.96e-2	1.45	367.52	1.28e-1	4.16
MLE	Nelder-Mead	165.63	4.61e-2	3.01	224.79	5.34e-2	54.98	357.53	1.4e-1	4.81
	Powell	165.62	4.6e-2	3.01	224.79	5.34e-2	54.98	357.53	1.4e-1	4.81
	L-BFGS-B	165.62	4.61e-2	3.01	224.79	5.34e-2	54.98	357.53	1.4e-1	4.81

Optimization approaches	Optimization algorithms	J1			J2			J3		
		a	b	c	a	b	c	a	b	c
	TNC	165.63	4.6e-2	3.01	224.84	5.27e-2	51.52	357.53	1.4e-1	4.81
	COBYQA	165.63	4.61e-2	3.01	224.79	5.34e-2	54.98	357.53	1.4e-1	4.81
	SLSQP	165.63	4.61e-2	3.01	224.79	5.34e-2	54.97	357.53	1.4e-1	4.81
	trust-constr	165.63	4.61e-2	3.01	224.79	5.34e-2	54.98	357.53	1.4e-1	4.81

〈부록 4〉 Weibull S-Shape parameters for the J1, J2 and J3 datasets

Optimization approaches	Optimization algorithms	J1			J2			J3		
		a	b	c	a	b	c	a	b	c
LSM	trf	7479.08	3.3e-4	9.8e-1	224	1.94e-6	2.95	373.1	1.36e-2	1.48
	dogbox	446440.38	2.21e-5	6.05e-1	224	2.41e-4	1.87	373.1	1.36e-2	1.48
MLE	Nelder-Mead	224360953.98	1.49e-8	8.92e-1	90682894.77	1.49e-8	9.83e-1	357.87	1.38e-2	1.52
	Powell	595.72	4.84e-3	9.58e-1	224.01	8.34	3.06e-2	357.93	1.38e-2	1.52
	L-BFGS-B	794.3	3.83e-3	9.37e-1	224.27	1.7e-5	2.47	357.87	1.38e-2	1.52
	TNC	794.27	3.83e-3	9.37e-1	244.69	1.01e-3	1.5	357.87	1.38e-2	1.52
	COBYQA	794.26	3.83e-3	9.37e-1	227.3	2.71e-4	1.85	357.87	1.38e-2	1.52
	SLSQP	794.04	3.83e-3	9.37e-1	224.28	1.69e-5	2.47	357.87	1.38e-2	1.52
	trust-constr	794.19	3.83e-3	9.37e-1	224.27	1.65e-5	2.48	357.87	1.38e-2	1.52

〈부록 5〉 Log-Logistic S-Shape parameters for the J1, J2 and J3 datasets

Optimization approaches	Optimization algorithms	J1			J2			J3		
		a	b	c	a	b	c	a	b	c
LSM	trf	12600.60	1.65e-4	9.81e-1	240.92	1.3e-2	3.76	460.31	5.68e-2	1.63
	dogbox	248647.36	4.96e-6	9.28e-1	240.92	1.3e-2	3.76	460.31	5.68e-2	1.63
MLE	Nelder-Mead	1944.94	9.64e-4	9.27e-1	66863887.27	1.49e-8	9.83e-1	422.07	6.43e-2	1.64
	Powell	1146.1	1.98e-3	9.67e-1	261.41	5.46e-1	3.90e-1	422.45	6.43e-2	1.64
	L-BFGS-B	1947.65	9.62e-4	9.27e-1	242.592	1.31e-2	2.89	422.08	6.43e-2	1.64
	TNC	1947.31	9.63e-4	9.27e-1	242.59	1.31e-2	2.89	422.08	6.43e-2	1.64
	COBYQA	1947.32	9.62e-4	9.27e-1	242.59	1.31e-2	2.89	422.08	6.43e-2	1.64
	SLSQP	1954.44	9.58e-4	9.27e-1	242.59	1.31e-2	2.89	422.08	6.43e-2	1.64
	trust-constr	1947.63	9.62e-4	9.27e-1	242.59	1.31e-2	2.89	422.08	6.43e-2	1.64

IDAP 광운대학교 방위사업연구소 INSTITUTE OF DEFENSE ACQUISITION PROGRAM

JAMS JOURNAL OF ADVANCES IN MILITARY STUDIES

2025, Vol. 8, No. 1, 15-36.

https://doi.org/10.37944/jams.v8i1.275

머신러닝 기반의 소프트웨어 신뢰성 성장 모델(SRGM) 최적화

신명근* · 정주원** · 이지현*** · 류인수**** · 박상건*****

국문초록

소프트웨어 신뢰성은 시스템의 성능 및 안전성 확보에 핵심 요소로 높은 신뢰성이 요구되는 방위산업 분야에서 그 중요성이 더욱 강조된다. 이에 현재 방위산업 분야는 소프트웨어 신뢰성 성장 모델(Software Reliability Growth Model, SRGM) 활용하여 소프트웨어 신뢰성을 효과적으로 평가하고 개선하고 있다. 그러나 선행연구는 주로 단일 최적화 기법이나 딥러닝 모델에 의존하여 국부 최적해(Local Optimum)에 수렴하거나 외삽 문제(Extrapolation Issue)로 인해 예측 신뢰성이 저하되는 한계를 보였다. 본 연구는 이러한 문제해결을 위해 최소자승법(Least Squares Method, LSM)과 최대우도법(Maximum Likelihood Estimation, MLE)을 기반으로 다양한 최적화 알고리즘을 적용하고 그 성능을 비교하여 전역 최적해(Global Optimum)에 근접한 결과를 도출하는 방안을 제시한다. 이 연구는 NASA JPL(Jet Propulsion Laboratory)의 소프트웨어 결함 데이터를 활용하였으며, 대표적인 SRGM 모델(Goel-Okumoto, Delayed S-Shape, Inflection S-Shape, Weibull, Log-Logistic)을 적용하여 최적화를 수행하였다. 연구 결과, 동일한 데이터셋에서도 최적화 방식과 알고리즘에 따라 예측 성능(Mean Squared Error, MSE)에 유의미한 차이가 발생하였다. 특히 J2 데이터의 Weibull 모델은 MSE가 70.778에서 15767.68까지 약 222배 이상의 차이를 보여 최적화 방법이 예측 정확도에 영향을 미쳤음을 확인할 수 있다. 따라서 본 연구는 소프트웨어 신뢰성 평가의 성능 향상을 위한 다양한 최적화 기법 활용의 방법론을 제시하였으며, 이를 통해 방산 분야에서 신뢰성 높은 소프트웨어 개발 및 평가에 실무적인 기여도가 있다고 볼 수 있다.

주제어 : 소프트웨어 신뢰성, 소프트웨어 신뢰성 성장 모델, 인공지능 최적화, 기계 학습

* (제1저자) ㈜모아소프트(고려대학교, 공학대학원), 선임 연구원(석사과정), mgshin@moasoftware.co.kr, https://orcid.org/0009-0001-3351-4580.

** (공동저자) ㈜모아소프트(광운대학교, 방산AI로봇융합학과), 선임 연구원(석사과정), jwjung@moasoftware.co.kr, https://orcid.org/0009-0007-4192-2713.

*** (공동저자) ㈜모아소프트(광운대학교, 방산AI로봇융합학과), 책임 연구원(석사과정), jhlee@moasoftware.co.kr, https://orcid.org/0000-0001-8165-2657.

**** (공동저자) ㈜모아소프트, AI/Data Science 연구소, 수석 연구원, isryu@moasoftware.co.kr, https://orcid.org/0009-0002-3215-0102.

***** (교신저자) ㈜모아소프트(광운대학교, 국방AI봇융합학과), 책임 연구원(박사과정), sgpark@moasoftware.co.kr, https://orcid.org/0009-0001-3196-510X.