Optimum Feature Selection in Software Product Lines: Let Your Model and Values Guide Your Search

Abdel Salam Sayyad Joseph Ingram Tim Menzies Hany Ammar
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV, USA
asayyad@mix.wvu.edu jingram3@mix.wvu.edu tim@menzies.us hany.ammar@mail.wvu.edu

***Abstract*—In Search-Based Software Engineering, well-known metaheuristic search algorithms are utilized to find solutions to common software engineering problems. The algorithms are usually taken "off the shelf" and applied with trust, i.e. software engineers are not concerned with the inner workings of algorithms, only with the results. While this may be sufficient is some domains, we argue against this approach, particularly where the complexity of the models and the variety of user preferences pose greater challenges to the metaheuristic search algorithms. We build on our previous investigation which uncovered the power of Indicator-Based Evolutionary Algorithm (IBEA) over traditionally-used algorithms (such as NSGA-II), and in this work we scrutinize the time behavior of user objectives subject to optimization. This analysis brings out the business perspective, previously veiled under Pareto-collective gauges such as Hypervolume and Spread. In addition, we show how slowing down the rates of crossover and mutation can help IBEA converge faster, as opposed to following the higher rates used in many other studies as "rules of thumb".**

***Index Terms*—Software Product Lines, Feature Models, Optimal Feature Selection, Multiobjective Optimization, Indicator-Based Evolutionary Algorithm, Search-Based Software Engineering.**

I. INTRODUCTION

Software Engineering technologies are increasingly having a direct impact on business outcomes, so much so that software decisions must be value-driven early on, i.e. business concerns ought to play a central role in the selection of software tools, technologies and processes. Barry Boehm states that: "software has a major influence on most systems' cost, schedule, and value; and value-neutral software decisions can seriously degrade project outcomes." [4] This comment is most relevant to the subject of this paper, in which we choose our evolutionary optimization algorithm to best exploit the user preferences in search for an optimal feature selection in a software product line.

Many software engineering researchers who seek to apply metaheuristic search methods to their problems choose the algorithms based on popularity of the algorithm in the SE literature, or its wide usage in other fields. Mark Harman makes the following comment regarding the choice of evolutionary search algorithms over non-evolutionary ones: "We must be wary of the unquestioning adoption of evolutionary algorithms merely because they are popular and widely applicable or because, historically, other researchers have adopted them for SBSE problems; none of these are scientific motivations for adoption." [9] We believe the same comment applies to choosing certain evolutionary algorithms over others.

In our previous work, we commented on the value of user preferences in search-based software engineering problems [12], where we concluded that the choice of which evolutionary algorithm to apply should follow the degree of complexity of the problem. High complexity in the decision space and the objective space require an algorithm that takes full advantage of the user preferences and we found that Indicator-Based Evolutionary Algorithm (IBEA) fit that bill. Compared to six other algorithms, including NSGA-II and SPEA2, IBEA was the only one that delivered satisfactory results when challenged with a complex model and four or five optimization objectives. The defining factor behind this is the way IBEA exploits the user preferences in computing the fitness assignment of the candidate solutions, as opposed to other algorithms that prioritize diversity of solutions in the fitness ranking. This will be briefly explained in section III, subsection D.

In this work, we build on the previous result in two important ways. First, we track the development of the user preferences over time to show the objectives as they get optimized by the metaheuristic algorithm. We present this as a way to demonstrate the benefit of the optimization method to the business user, as we show the way multiple objectives play trade-offs against one another while they get optimized together in the Pareto sense. Second, we show an order-of-magnitude improvement of IBEA via tuning down the crossover and mutation rates. The "rule-of-thumb" rates often used with evolutionary algorithms (including our own previous work) amount to lengthening the duration of the search. This highlights the fine-grained structure of the feature models, where small changes in features have great effects on other features, and thus the search needs to proceed slowly for better exploration of the decision space.

The rest of the paper is organized as follows: section II reviews related work in automated software product configuration. Section III provides background material on software product lines and MEOAs. Sections IV and V

describe the experimental setup and the results of the experiments. In sections VI, we discuss the findings and their implications. We present our conclusion in section VII.

II. Related Work

Here we discuss related work in the area of automated product configuration and feature selection.

The idea of extending (or augmenting) feature models with quality attributes was proposed by many, among them Zhang et al. [17]. The following papers used a similar approach and attached synthetic data to represent the attributes in SPLs for the sake of experimentation, as did we in this study and the previous one [12].

The following non-search-based methods were used in earlier studies:

- Constraint Satisfaction Problem (CSP) solvers, used by Benavides et al. [3] and White et al. [16].
- Filtered Cartesian Flattening used by White et al. [15].
- Hierarchical Task Network planning used by Soltani et al. [13].

More recently, a Genetic Algorithm was used by Guo et al. to tackle this problem [8]. Although the problem is obviously multiobjective, the various objectives where aggregated into one and a simple GA was used. The result was a single "optimal" configuration, which is only optimal according to the weights chosen in the objective formula. Also, they used a repair operator to keep all candidate solutions in line with the feature model all throughout the evolutionary process.

In our own previous study [12], we employed Multiobjective Evolutionary Optimization Algorithms (MEOAs) for the first time to address this problem, with a maximum of 5 objectives. We treated "correctness" as one of the optimization objectives, where the MEOAs were charged with minimizing rule violations. We compared performances of 7 algorithms and showed that IBEA outperformed all others, including NSGA-II and SPEA2.

In this study, we explore the behavior of various objectives throughout the evolution process, and we demonstrate faster convergence for IBEA when lower values are chosen for crossover and mutation rates.

III. Background

A. Feature Models for Software Product Lines

A feature is an end-user-visible behavior of a software product that is of interest to some stakeholder. A feature model represents the information of all possible products of a software product line in terms of features and relationships among them. A feature model encompasses a hierarchically arranged set of features composed by:

1- Relationships between a parent feature and its child features (or subfeatures).
2- Cross-tree constraints that are typically inclusion or exclusion statements in the form: if feature F is included, then features A and B must also be included (or excluded).

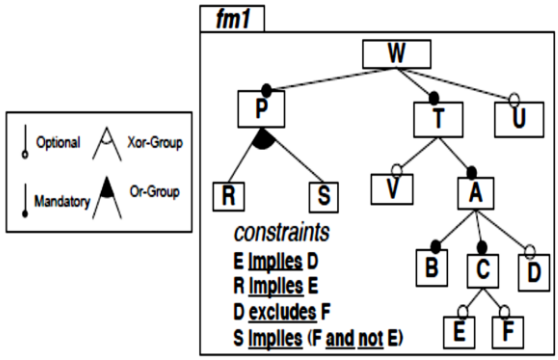

Figure 1: Example feature model

Figure 1, adopted from [1], depicts a simplified feature model.

The full set of rules in a given feature model may include the following:

- The root feature is mandatory.
- Every child requires its parent.
- If the child is mandatory, the parent requires the child.
- Every group adds a rule about how many members can be chosen.
- Every cross-tree constraint (CTC) is a rule.

Thus it can be concluded that the feature model depicted in Figure 1 includes a total of 23 rules.

In this experiment, the total number of rules is used as the "full correctness" score in this experiment, thus making "correctness" one of the optimization objectives.

B. Multiobjective Optimization

Many real-world problems involve simultaneous optimization of several incommensurable and often competing objectives. Often, there is no single optimal solution, but rather a set of alternative solutions. These solutions are optimal in the wider sense that no other solutions in the search space are superior to them when all objectives are considered [19].

Formally, a *vector* $u = \{u_1, \ldots, u_k\}$ is said to dominate a vector $v = \{v_1, \ldots, v_k\}$ if and only if u is partially less than v, i.e.

$$\forall i \in \{1, \ldots, k\}, u_i \leq v_i \; and \; \exists \, i \in \{1, \ldots, k\} : u_i < v_i \quad (1)$$

The set of all points in the objective space that are not dominated by any other points is called the *Pareto Front*.

C. Multiobjective Evolutionary Optimization Algorithms (MEOAs)

The algorithms we used in this study are implemented in the jMetal framework [6]. They are:

1- NSGA-II: Nondominated Sorting Genetic Algorithm, version 2 [5].
2- SPEA2: Strength Pareto Evolutionary Algorithm, version 2 [20].
3- IBEA: Indicator-Based Evolutionary Algorithm [18].

D. Ranking Criteria in MEOAs

All three MEOAs used in this study are based on Genetic Algorithms. They share some basic qualities, such as: single-point crossover, bit-flip mutation, binary tournament for mating selection, and elitism. They also have differences, the most relevant to mention here being the ranking criterion (i.e. fitness assignment) used to determine which individuals have stronger chance to survive to the next generation. Those criteria are:

1- NSGA-II: The sorting procedure in NSGA-II is depicted in Figure 2, taken from [5]. It shows how the combined primary and secondary population gets sorted according to domination, where F_1 contains all nondominated solutions; F_2 contains all nondominated solutions after excluding F_1 and so on. When the solutions within F3 need to be sorted for truncation, they are ranked according to crowding distance, a value calculated from distances to nearest neighbors in all objective values. Thus diversity preservation is the second criterion –after domination- to determine fitness for survival.

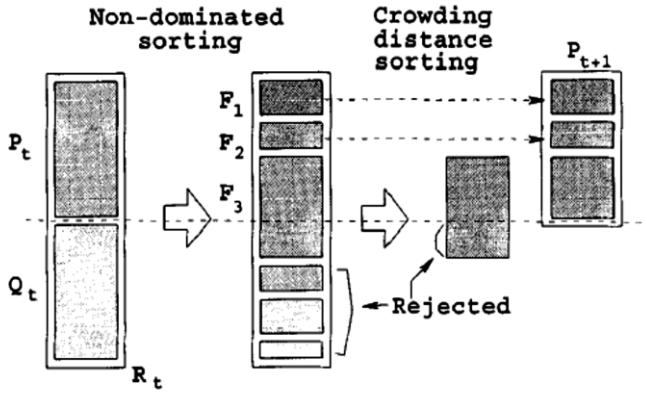

Figure 2: NSGA-II sorting procedure

2- SPEA2: The sorting procedure in SPEA2 is somewhat similar to that depicted in Figure 2, with two differences. First, domination sorting only takes place once, thus dividing the population into F_1 and F_2. Second, the ranking criterion for individuals in F_2 is based on the *strength* of each solution, defined as the number of solutions that are dominated by it. The fitness value of a point is the sum of strengths of all solutions that dominate that point added to a *density estimation* that works to prioritize points with less proximity to nearest neighbors.

3- IBEA: Equation 2 shows IBEA's fitness assignment.

$$F(x_1) = \sum_{x_2 \in P \setminus \{x_1\}} -e^{-I(\{x_2\},\{x_1\})/\kappa} \qquad (2)$$

Each solution is given a weight based on *I(.)*, a dominance-preserving quality indicator, thus factoring in more of the optimization objectives of the user. The authors of IBEA, Zitzler and Kunzli, designed the algorithm such that "preference information of the decision maker" can be "integrated into multiobjective search" [18]. It is noticed here that the ranking criteria in IBEA place no emphasis on diversity of solutions, thus diverging from the conventional trend set by NSGA-II and SPEA2, and followed by many others.

This difference in ranking criteria causes IBEA to outperform NSGA-II and SPEA2 when the objective space increases in dimension. In [14], it is experimentally demonstrated with real-valued test functions that the performance of NSGA-II and SPEA2 rapidly deteriorates with increasing dimension, and that other algorithms like ε-MOEA, MSOPS, IBEA and SMS-EMOA cope very well with high-dimensional objective spaces. It is argued that NSGA-II and SPEA2 tend to "increase the distance to the Pareto front in the first generations because the diversity-based selection criteria favor higher distances between solutions. Special emphasis is given to extremal solutions with values near zero in one or more objectives. These solutions remain non-dominated and the distance cannot be reduced thereafter."

E. Quality of Pareto Front

We compare the performance of MEOAs using the following quality indicators:

1- Hypervolume (HV): defined in [19], is a measure of the size of the space covered underneath the Pareto front. If the objectives are all to be maximized, then the preferred Pareto front is the one with the highest Hypervolume. In jMetal, all objectives are minimized, but the Pareto front is inverted before calculating hypervolume, thus the higher the hypervolume the closer to optimum the Pareto front is.

2- Spread: defined in [5], measures the extent of spread in the obtained solutions.

3- %correct: i.e. the percentage of fully-correct solutions, which is an indicator particular to this problem. Since correctness is an optimization objective that evolves over time, there maybe points in the final Pareto front that have rule violations. Such points are not likely to be useful to the user. We are interested in percentage of points within the Pareto front that have zero violations, and thus a full-correctness score.

IV. Setup

A. Setting Up the E-Shop Feature Model

The E-Shop feature model is the largest member of the feature model repository at SPLOT website [11]. It consists of 290 features, 21 CTCs, and a total of 421 rules.

We augmented the feature model with 3 attributes per feature: *COST*, *USED_BEFORE*, and *DEFECTS*. COST takes real values distributed normally between 5.0 and 15.0, USED_BEFORE takes Boolean values distributed uniformly, and DEFECTS takes integer values distributed normally between 0 and 10. The only dependency among these qualities is:

if (not USED_BEFORE) then DEFECTS = 0 (3)

B. *Problem Encoding*

The feature models were represented as binary strings, where the number of bits is equal to the number of features. If the bit value is *TRUE* then the feature is selected, otherwise the feature is removed.

C. *Defining the Optimization Objectives*

In this work we optimize the following objectives:

1- Correctness; i.e. compliance to the relationships and constraints defined in the feature model. Since jMetal treats all optimization objectives as minimization objectives, we seek to minimize rule violations.

2- Richness of features; we seek to minimize the number of deselected features.

3- Features that were used before; we seek to minimize the number of features that weren't used before.

4- Known defects; which we seek to minimize.

5- Cost; which we seek to minimize.

V. RESULTS

The experiment is divided into three parts. In the first part, we run IBEA over the augmented E-Shop feature model using the same parameter values as in our previous work [12], and we plot the quality indicators and the normalized mean objective values over time. In the second part, we make the same two plots with reduced values for crossover rate and mutation rate. And in the third part, we explore the effect of reducing the crossover and mutation rates on the performance of all three MEOAs (IBEA, NSGA-II, and SPEA2).

A. *IBEA Performance Over Time*

In this part, we run IBEA with the same parameters that we used in our previous work [12]. This includes a crossover rate of 0.9 and a mutation rate of 0.05. The measurements are taken over 5 hours of evolution.

First, we're interested in the development of the quality indicators over time, which is plotted in Figure 3. We make the following observations:

1- The %correct indicator does not show any fully-correct solutions until after 10 minutes of operation. After 5 hours, there are 42 solutions out of a 100 members of the Pareto front that are fully compatible with the feature model.

2- The Hypervolume (HV) indicator continues growing as we edge closer to optimality, but the growth is rather slow. The HV value at 3 hours is 99% of that achieved at 5 hours.

3- The Spread varies over the period of operation, and reaches its highest values while the solutions are suboptimal and highly inconsistent with the feature model.

Next, in Figure 4, we plot the normalized mean values for each of the objectives over time. Since all the objectives are to be minimized, plotting the mean values is expected to show a trend towards minimum values. We observe the following:

1- Four of the five objectives trend downward, until they reach their least values at the end of operation. The trend is interrupted with swings upward, as the objectives play trade-offs against one another along the way.

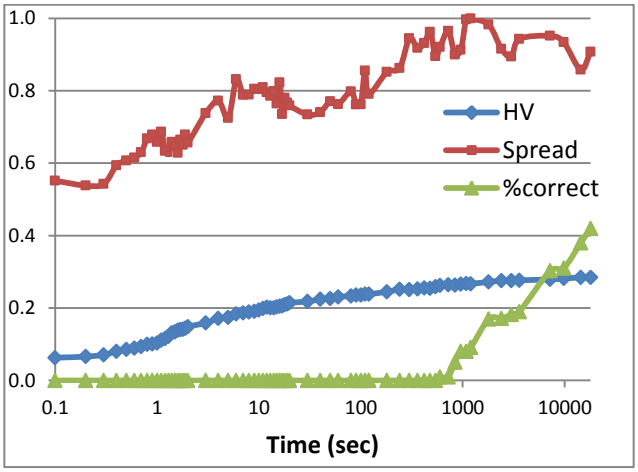

Figure 3: Quality indicators vs. time

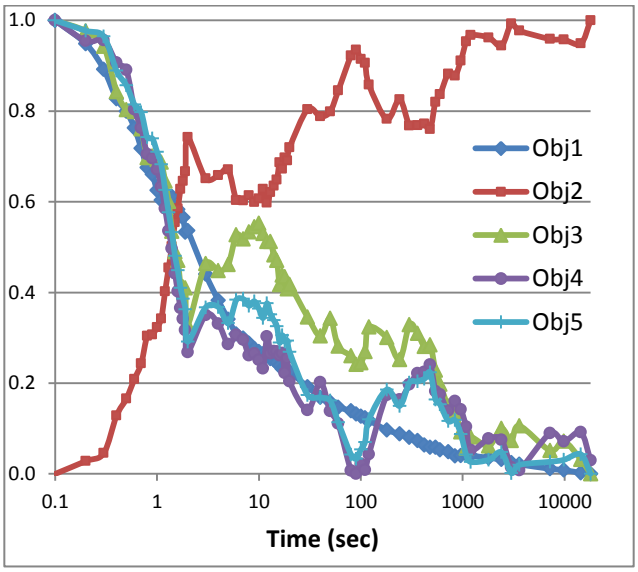

Figure 4: Normalized mean objective values vs. time

2- The second objective –number of missing features- trends upward, until it reaches its highest value at the end of operation, despite being defined as a minimization objective. The observation here is: there is an expected correlation between having more features and higher cost, defects, and features not used before. Therefore, the multiobjective optimization process finds it best to push the missing features upward (total features downward) in order to achieve best overall Pareto optimality.

B. *Reducing the Rates of Crossover and Mutation*

We now tweak the genetic operator parameters downward; we use a crossover rate of 0.1 and a mutation rate of 0.01. We stop the operation after 1 hour since we see an early plateau in the indicators.

The quality indicators are plotted in Figure 5, and the observations are:

1- The %correct indicator begins showing 74% fully-correct solutions after 12 minutes of operation. After 1 hour, the percentage stands at 75%.

2- At 12 minutes, the HV indicator shows 99% of its final value at the end of the operation. This tells us that the continuing optimization after 12 minutes is a waste of CPU power in return for insignificant gain. In part A above, HV achieved its 99% after 3 hours. Thus the run time improvement achieved by reducing the parameters is 15 folds, i.e. a whole order of magnitude.

3- The Spread indicator reaches high values early in the operation, when the solutions are suboptimal and inconsistent with the feature model. The Spread values achieved after 12 minutes represent the nominal diversity measurements for solutions whose majority fully-conforms to the feature model.

As for the normalized mean objective values, plotted in Figure 6, we observe similar trends as those mentioned in part A; objective 2 trends upward as other objectives get minimized. The main reason for this trend is correctness; the more features included in configurations the more constraint violations there would be. The algorithm learns to include fewer features as it achieves full correctness.

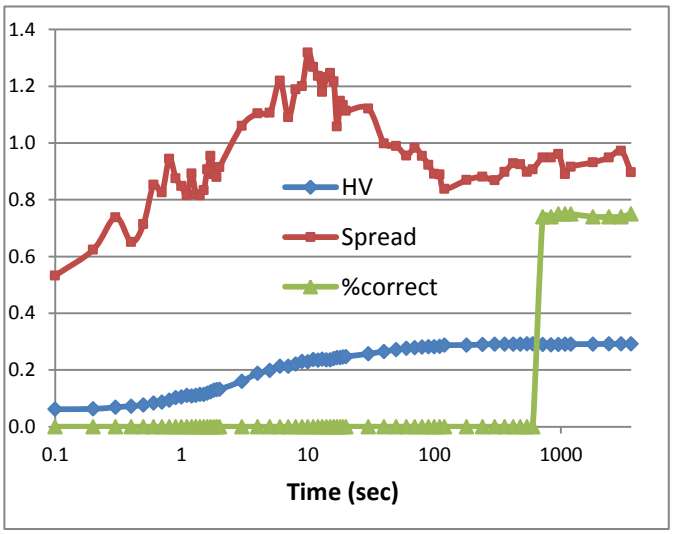

Figure 5: Quality indicators vs. time

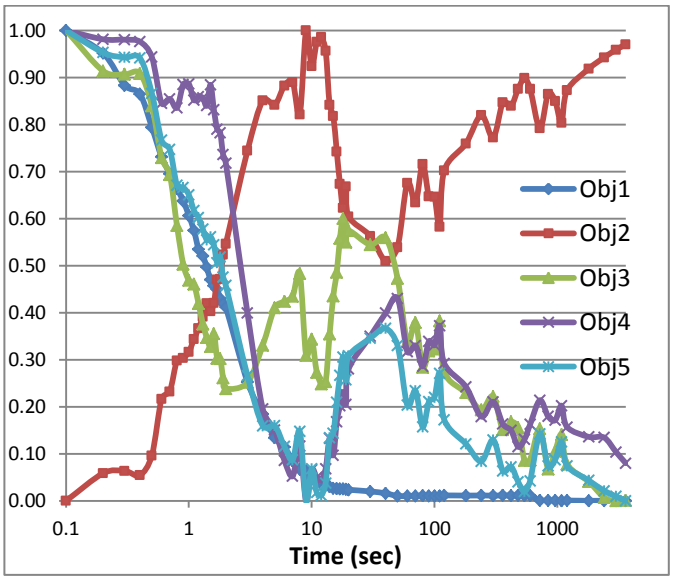

Figure 6: Normalized mean objective values vs. time

C. *Comparing MEOAs with High and Low Rates of Crossover and Mutation*

In this part, we run a large experiment, in which we seek to compare the performance of the 3 MEOAs with varying parameters, according to Table 1. Each method is run 10 times, each time for duration of 30 minutes.

TABLE 1: OUTLINE OF EXPERIMENT IN PART C

Method	Crossover rate	Mutation rate	Duration	Runs
IBEA-h	0.9	0.05	30 min	10
IBEA-l	0.1	0.01	30 min	10
NSGA2-h	0.9	0.05	30 min	10
NSGA2-l	0.1	0.01	30 min	10
SPEA2-h	0.9	0.05	30 min	10
SPEA2-l	0.1	0.01	30 min	10

In Table 2, we show the resulting mean values and standard deviations of the quality indicators. The methods are sorted according to Hypervolume (HV). The "Effect size" column shows Hedge's effect size which is computed for the HV of each method versus the method below it. The description of the effect as large follows the classification in Table 9 of Kampenes et al. [10].

TABLE 2: RESULTS OF EXPERIMENT IN PART C

Method	HV-mean	HV-stdev	Effect Size	%correct mean	Spread mean
IBEA-l	0.293	0.0016	7.66 (large)	66.8%	0.89
IBEA-h	0.271	0.0033	13.35 (large)	9.9%	0.88
NSGA2-h	0.211	0.0047	1.57 (large)	0.6%	0.78
SPEA2-l	0.204	0.0032	1.14 (large)	0.8%	0.63
NSGA2-l	0.192	0.0130	1.58 (large)	2.4%	0.95
SPEA2-h	0.174	0.0066	--	0.0%	0.56

These results show the following:

1- IBEA with low parameters performs remarkably better than all, both in terms of HV and %correct, followed by IBEA with high parameters which also beats all others by a large margin in both HV and %correct.

2- The highest spread value was achieved by NSGA-II with low parameters, which indicates a better diversity of results. Nevertheless, since it falls short on optimality (measured with HV) and correctness, the solutions are useless. In fact, IBEA achieves very good spread values, both at high and low parameters.

VI. DISCUSSION

First, we would like to comment on the different viewpoints offered by quality indicators (Figures 3 & 5) and the normalized mean values (Figures 4 & 6). MEOAs are usually compared with the help of quality indicators, since they offer an aggregate evaluation of the entire Pareto front. This viewpoint is useful for researchers, but not so for business end-users. When the objectives are spelled out and plotted alongside one another in the form of normalized mean values, the user can realize the benefit of multiobjective optimization with IBEA.

Reducing the crossover rate and mutation rate has significantly improved the performance of IBEA. In fact, the run time savings in this paper over our previous work [12] is 15 folds. This hints at the fine-grained structure of the feature models. Low rates of crossover and mutation promote moving slowly through the feature model to discover better solutions, whereas high rates promote vast changes to the individuals from generation to generation.

This reduction in the parameters goes against the common rule of thumb. For example, Eiben and Smith [7] suggest a crossover rate between 0.6 and 0.9. These suggestions are usually taken without question. Arcuri and Fraser [2] showed that the choice of parameter values can result in large variances in performance of evolutionary algorithms.

For NSGA-II and SPEA2, reducing those parameters didn't help much, and we still obtained solutions that are inconsistent with the feature model. This is due to the diversity preservation measures in both algorithms which tend to disallow the crowding of solutions close to the much desired zero-violation point.

VII. Conclusion

It is tempting for researchers and industrial practitioners to apply optimization tools using their off-the-shelf parameters and study their results using standard performance measures such as hypervolume and spread. While this approach may sometimes work in domains with only 1 or 2 objectives, we strongly discourage that practice for problems with a rich set of objectives. In [12] we showed the superiority of IBEA in searching complex structures with many objectives. In this paper we tracked the development of the optimization objectives over time, highlighting the importance of this tracking to the business end-user. Furthermore, we demonstrated an order-of-magnitude improvement in the performance of IBEA via tuning down the rates of the genetic operators. Such enhancement in performance followed from the nature of the models, which required small changes across generations for a better exploration of possible configurations.

Acknowledgment

This research work was funded by the Qatar National Research Fund (QNRF) under the National Priorities Research Program (NPRP) Grant No.: 09-1205-2-470.

Joseph Ingram's work was supported by National Science Foundation (NSF) Grant No.: CCF 1017330.

References

[1] M. Acher, P. Collet, P. Lahire, and R. France, "Decomposing Feature Models: Language, Environment, and Applications," in *Proc. ASE*, Lawrence, KS, USA, 2011, pp. 600-603.

[2] A. Arcuri and G. Fraser, "Parameter tuning or default values? An empirical investigation in search-based software engineering," *Empirical Software Engineering*, February 2013.

[3] D. Benavides, A. Ruiz-Cortés, and P. Trinidad, "Automated Reasoning on Feature Models," in *Proc. CAISE*, 2005, pp. 491-503.

[4] B. Boehm, "Value-Based Software Engineering," *Software Engineering Notes*, vol. 28, no. 2, pp. 1-12, March 2003.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[6] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760-771, 2011.

[7] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*.: Springer, 2003.

[8] J. Guo, J. White, G. Wang, J. Li, and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *Journal of Systems and Software*, vol. 84, no. 12, pp. 2208–2221, December 2011.

[9] M. Harman, "Software Engineering Meets Evolutionary Computation," *IEEE Computer*, vol. 44, no. 10, pp. 31-39, October 2011.

[10] V. B. Kampenes, T. Dyba, J. E. Hannay, and D. I.K. Sjøberg, "A systematic review of effect size in software engineering experiments," *Information and Software Technology*, no. 49, pp. 1073–1086, 2007.

[11] M. Mendonca, M. Branco, and D. Cowan, "S.P.L.O.T. - Software Product Lines Online Tools," in *Proc. OOPSLA*, Orlando, USA, 2009.

[12] A. S. Sayyad, T. Menzies, and H. Ammar, "On the Value of User Preferences in Search-Based Software Engineering: A Case Study in Software Product Lines," in *Proc. ICSE*, San Francisco, USA, 2013, p. To appear.

[13] S. Soltani, M. Asadi, H. Marek, D. Gasevic, and E. Bagheri, "Automated Planning for Feature Model Configuration based on Stakeholder's Business Concerns," in *Proc. ASE*, Lawrence, KS, USA, 2011, pp. 536-539.

[14] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization," in *Proc. EMO, LNCS Volume 4403/2007*, 2007, pp. 742-756.

[15] J. White, B. Dougherty, and D. C. Schmidt, "Selecting highly optimal architectural feature sets with Filtered Cartesian Flattening," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1268–1284, August 2009.

[16] J. White, B. Dougherty, D. C. Schmidt, and D. Benavides, "Automated reasoning for multi-step feature model configuration problems," in *Proc. SPLC*, San Francisco, USA, 2009, pp. 11-20.

[17] G. Zhang, H. Ye, and Y. Lin, "Using Knowledge-Based Systems to Manage Quality Attributes in Software Product Lines," in *Proc. SPLC*, 2011.

[18] E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 2004, pp. 832–842.

[19] E. Zitzler and Thiele L., "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[20] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," in *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*. Athens, Greece, 2001, pp. 95-100.