SEARCH BASED SOFTWARE ENGINEERING (SBSE) A SURVEY

Mohamad Syafri Tuloli[1)], Benhard Sitohang [2)], Bayu Hendradjaya [3)]

[1), 2),3)] School of Electrical Engineering & Informatics - ITB
Jl. Ganesa No. 10, BANDUNG-40132
Email : syafri_tuloli@students.itb.ac.id[1)], benhard@stei.itb.ac.id *[2)],* bayu@informatika.org *[3)]*

Abstrak

Penerapan Search-Based Optimization pada Software Engineering berkembang dengan pesat dalam beberapa tahun terakhir. Paper ini membahas mengenai topik-topik penelitian, metode-metode optimasi yang sering digunakan, kelebihan, kendala dan solusi dalam penerapan, serta potensi pengembangan lebih lanjut dari penelitian pada bidang Search-Based Opmitization untuk Software Engineering. Survey yang dilakukan juga mencoba melihat tren paper-paper bidang SBSE dilihat dari metode, fase software engineering, dan eksplorasi yang telah dilakukan dalam arah-arah potensi pengembangan.

Kata kunci: *Software engineering, search-based optimization, survey.*

1. Pendahuluan

Dalam proses rekayasa suatu perangkat lunak (Software Engineering) hampir selalu terdapat permasalahan yang berkaitan dengan optimasi, masalah seperti pemilihan dan penentuan prioritas kebutuhan, penentuan test case, penentuan modularisasi kelas, dan lain-lain. Permasalahan-permasalahan optimasi tersebut khususnya yang memiliki kompleksitas cukup tinggi sangat cocok untuk diselesaikan menggunakan metode berbasis pencarian (*search-based*). Hal ini karena untuk mengimplementasikan metode SBSE hanya dibutuhkan dua hal yaitu representasi permasalahan dan dan definisi **fungsi fitness** (fungsi untuk mengevaluasi solusi) [1], sedangkan dalam permasalahan biasanya sudah terdapat representasi permasalahannya dan untuk fungsi fitness bisa menggunakan metrik yang sudah banyak terdapat rekayasa perangkat lunak [2].

Pembahasan yang dilakukan dalam paper ini adalah tentang penerapan metode optimasi (berbasir pencarian) dan aspek umum dari penerapan pada bidang rekayasa perangkat lunak (*software engineering*). Pada bagian 2, dibahas mengenai metode metaheuristik dan beberapa contoh metode yang sering digunakan serta kelebihan, kendala umum dan khusus dari penerapan metode optimasi metaheuristik ini pada perangkat lunak secara umum. Arah Penelitian dari survey sebelumnya yang dijelaskan pad abab 3. Selunjutnya untuk hasil dari survey dan analisis dari hasil yang didapatkan dibahas pada bab 4. Bab 5 membahas arahan ke depan yang diusulkan oleh paper ini berdasarkan hal dirasa yang dibutuhkan untuk dikembangkan. Bab 6 menjelaskan kekurangan dan Threat terhadap validitas dari hasil yang didapatkan baik dari metode yang digunakan, pengetahuan penulis, maupun bias yang mungkin terjadi. Bab 7 dan bab 8 membahas kesimpulan dari hasil survey yang dilakukan dan upaya pengembangan selanjutnya dari riset yang dilakukan.

Search Based Software Engineering

A. MetaHeuristic

Heuristik bersal dari bahasa Yunani heuriskein yang berarti "mencari", metode heuristic sendiri berarti algoritma approksimasi untuk mencari solusi yang baik pada ruang solusi (ruang semua kemungkinan solusi, sering disebut ruang pencarian) [3]. Meta berasal dari bahasa Yunani yang berarti "melebihi, atau tingkat atas". Algoritma metaheuristik adalah algoritma ynag mengkombinasikan heuristic (yang biasanya sangat problem-specific) pada framework yang lebih umum agar pencarian solusi near-optimal menjadi lebih efisien [4].

Tidak seperti heuristic, metaheuristik tidak problem-dependent sehingga bisa diaplikasikan pada banyak domain permasalahan. Metaheuristik juga bersifat non-deterministik sehingga tidak menjanjikan solusi yang optimal, tetapi hanya paling near-optimal (dekat dengan optimal). Metode metaheuristik dapat dibagi berdasarkan sifat-sifatnya [4] :

1). Nature-inpsired dan non-nature-inspired
Beberapa metode secara jelas terinspirasi dari alam seperti Algoritma Genetika, Algoritma semut, Simulated Annealing, Hill Climbing, dll, sedangkan metode lain yang *non-nature-inspired* seperti Tabu Search.
2). Berbasis populasi dan pencarian satu titik
Dalam melakukan pencarian ada metode-metode yang menggunakan satu solusi atau satu titik pada ruang pencarian, metode jenis ini sering disebut trajectory-methods, contohnya adalah Tabu search, hill climbing, iterated local search, variable neighborhood. Lawannya adalah metode yang menggunakan sekumpulan solusi (populasi) seperti Algoritma Genetika.
3). Fungsi objektif yang statik dan dinamis
Beberapa metode menggunakan fungsi objective tanpa perubahan, beberapa metode lain (mis. Guided Local Search) mengubah fungsi objectif sehingga juga ruang

pencarian, tujuannya adalah untuk keluar dari local optima.

4). Menggunakan memory dan memory-less

Penggunaan memory dibagi lagi menjadi menggunakan memory jangkap pendek dan memory jangka panjang. Penggunaan memory jangka pendek biasanya untuk menyimpan keputusan yang diambil (mis. langkah yang sudah pernah dilakukan atau solusi yang pernah dikunjungi). Memory jangka panjang biasanya untuk menyimpan akumulasi nilai parameter selama melakukan pencarian.

B. *Search Based Optimization*

Search Based search adalah penggunaan metode-motoda optimasi, khususnya metode metaheuristik. Keuntungan-keuntungan yang didapatkan dengan menggunakan SBSE [5][6]:

1). Generality

Sangat mudah diaplikasikan pada berbagai permasalahan, dengan hanya membutuhkan representasi dari permasalahan dan fungsi fitness. Sedangkan setiap permasalahan terhadap suatu permasalahan selalu dimulai dengan penentuan representasinya, dan fungsi fitness dapat diadaptasi dari metrik-metrik software engineering yang sudah banyak tersedia.

2). Robust

Parameter tuning akan membantu pencarian solusi, tetapi tanpa parameter tuning pun metode SBSE (dengan parameter yang tidak ekstrim) selalu lebih unggul dari metode pencarian yang murni random.

3). Scalability

Penghitungan nilai fungsi fitness pada SBSE sangat memungkinkan (dan telah digunakan) untuk dieksekusi secara paralel.

4). Direct Fitness Function

Penghitungan nilai fitness dari software yang akan dioptimasi, dilakukan secara langsung, tidak memerlukan simulasi dari model seperti pada optimasi bidang rekayasa lain.

5). Reunification

SBSE memunculkan hubungan/kesamaan antara permasalahan yang dihadapi antara daerah dalam software engineering yang sebelumnya dianggap tidak sama, misalnya permasalahan prioritisasi kebutuhan pada requirement engineering ternyata dapat dipecahkan dengan cara yang sama dengan permasalahan prioritisasi test case pada testing.

6). Feedback dan Insight

SBSE melakukan pencarian secara otomatis dan tidak terhalangi oleh bias seperti pada pencarian yang dilakukan oleh orang, dan seringkali menghasilkan solusi yang sangat tidak diduga. Hal ini membuat SBSE menjadi pelengkap dari pencarian secara manual, dimana orang dapat menyempurnakan fungsi fitness sehingga dapat menangkap asumsi dan intuisi pengguna dengan lebih baik.

Walaupun dengan kelebihan-kelebihan tersebut, penerapan SBSE bukan tanpa kendala, beberapa diantara kendala tersebut yang mungkin dihandapi seperti [6]:

1). Fungsi Fitness dengan biaya komputasi tinggi

Sebagian besar komputasi pada SBSE terjadi pada fungsi fitness, sehingga fungsi fitness yang tinggi harus disiasati dengan menggunakan fungsi pengganti, penghitungan fitness secara parallel atau mengeksploitasi pengetahuan domain permasalahan.

2). Fungsi fitness kurang bisa didefinisikan dengan jelas

Seringkali fungi fitness yang digunakan hanya berupa apploksimasi yang memberikan arah secara umum dan bisa jadi tidak dapat menilai sebagian kandidat solusi, tetapi bisa disiasati dengan menggunakna penilaian manusaia, atau menggunaan beberapa fungsi fitness sekaligus (mis. untuk waktu pencarian, daerah solusi, stakeholder, dll).

3). Search space yang terlalu sulit untuk dilakukan pencarian

Penggunaan fungsi fitness dapat mengalami memiliki kendala pada suatu daerah pencarian, sehinga mungkin bisa diatasi dengan menggunakan fungsi fitness alternative, menggunakan lebih dari satu fungsi fitness (yang setara atau bertingkat). Selain itu dapat juga menggunakan algoritma lainnya.

4). Fitness dari Solusi akhir yang didapatkan terlalu rendah

Hasil pencarian dapat memberikan hasil dengan nilai fitness yang rendah, diakibatkan pemilihan parameter atau algoritma yang kurang tepat. Untuk mengatasinya mungkin dengan mencoba algoritma lain, atau menggunakan algoritma yang berbeda untuk setiap daerah pencarian (kombinasi/memetic). Hasil yang dengan fitness rendah tetap masih bisa digunakan untuk mengenali karakteristik daerah pencarian dan karakteristik solusi yang optimal.

Permasalahan pada Rekayasa perangkat lunak menjadi domain yang sangat tepat untuk penerapan SBO [7](Search Based Optimization) dikarenakan metode SBO membutuhkan hanya dua hal saja, yaitu representasi permasalahan, dan fungsi fitness untuk mengukur suatu kandidat solusi. Sedangkan SE (Software Engineering) kaya akan metrik yang merupakan bahan utama untuk membuat fungsi fitness. Untuk menggunakan software metrics sebagai fungsi fitness, metriks tersebut sedikitnya memiliki karakteristik sebagai berikut [2]: memiliki Search space yang luas dan kontinyu, kompleksitas komputasi rendah, dan tidak diketahuinya solusi yang optimal.

Arah Penelitian dari Survey sebelumnya

Arah penelitian yang potensial pada bidang SBSE secara umum menurut survey-survey sebelumnya adalah [35] [5] [33] yang dapat dibagi menjadi dua arahan :

1). Arah metode

Penggunaan hybrid Optimisation adalah penggabungan metode-metode pencarian sehingga dapat memanfaatkan kelebihan masing-masing dalam mencari solusi yang optimal. Misalnya hill climbing pada setiap solusi yang dihasilkan dari metode GA, sehingga GA yang memiliki kelebihan mencari global optimum mendapakan keuntungan dari Hill Climbing yang walaupun bersifat lokal optimal tapi memiliki biaya komputasi yang murah.

Selain menggabungkan method, arah lain adalah dengan memecah-mecah solusi menjadi beberapa bagian. Ko-evolution yaitu optimasi yang menggunakan dua atau lebih populasi yang berevolusi dimana populasi saling mempengaruhi dan berhubungan secara kooperatif atau bisa juga secara kompetitif. Ide dasar coevolution adalah menggunakan strategi divide-and-conquer dengan melakukan evolusi terhadap solusi secara keseluruhan dan modul-modul dari solusi tersebut secara simultan [38], hal ini berdasarkan hipotesa bahwa pencarian secara paralel terhadap bagian-bagian dari solusi lebih efisien dari satu pencarian untuk satu kesatuan solusi dan bisa menghilangkan kemungkinan konvergensi ke satu solusi [39].

Permasalahan pada software engineering seringkali membutuhkan solusi yang dinilai dari pemenuhannya terhadap lebih dari satu ojbektif. Pendekatan MOO (*Multi-Objektif Optimization*) melakukan pencarian dengan mengasumsikan bahwa solusi yang optimal bukan berada pada satu titik, tetapi adalah suatu area yang disebut pareto front. Hal ini memungkinkan didapatkannya solusi-solusi terbaik dilihat dari beberapa objektif, sehingga pengguna dapat memilih solusi-solusi yang terbaik menurut objektif yang menjadi prioritasnya.

Dengan semakin banyaknya pilihan metode metaheuristik dan juga berkembangnya metode Genetic Programming melahirkan pemikiran untuk menggunakan keadaan ini dengan membuat metode untuk memilih metode ataupun metode untuk membuat metode yang disebut hyperheuristik. Hyper heuristic adalah metode dimana penerapan pencarian dilakukan pada tingkat heuristic, yaitu dengan melakukan pencarian heuristic yang optimal untuk memecahkan suatu permasalahan. Pendekatan ini berbeda dengan pendekatan metaheuristik biasa yang ruang-pencariannya (search-space) berupa seluruh kemungkinan solusi, ruang-pencarian hyperheuristic adalah seluruh kemungkinan heuristic yang dapat digunakan untuk pemecahan solusi.

Pendefinisian fungsi fitness sering menjadi tantangan tersendiri dalam penerapan SBSE, untuk permasalahan yang sulit didefinisikan fungsi fitnessnya (subjektif, menggunakan pengetahuan tacit, dll), dapat menggunakan pendekatan inteaktif yaitu penentuan nilai fitenesnya dilakukan oleh manusia/pakar. Tetapi harus diperhitungkan efek dari kelelahan dan bias *learning-effect.*

2). Arah permasalahan

Arah permasalahan yang diusulkan adalah lebih pada penerapan SBSE untuk analisis source code, hal ini juga didorong karena masih kurangya penerapan pada tahapan implementasi. Selain itu penerapan yang dianggap menarik adalah pada optimasi property non-fungsional dari perangkat lunak, pada protocol (efisiensi, sekuritas, biaya dan tingkat kebenaran). Arah lain adalah penerapan pada permasalahan yang dapat memanfaatkan prinsip-prinsip pada information theory sebagai fungsi fitnessnya.

Karena sebagian metode-metode metaheuristik yang digunkan adalah berbasis populasi, SBSE dianggap cocok untuk digunakan pada *software agent* dan *distributed intelligence.* Sekuritas dan Protection juga masih belum banyak dieksplorasi menggunakna SBSE karena tantangan yang dihadapi dalam mendefinisikan fungsi fitnessnya.

Beberapa metode metaheuristik memiliki sifat online optimization, yaitu dapat melakukan optimasi pada permasalahan yang terus berubah saat optimasi dilakukan. Penerapan SBSE pada permasalahan optimasi yang bersifat dinamis dan terus berubah ini, ini cocok menggunakan teknik optimasi seperti PSO (Particle Swarm Optimization) dan ACO (Ant Colony Optimization).

2. Survey Result dan Analysis

Melanjutkan klasifikasi yang digunakan oleh Hamann [5] dari paper tahun 2010-2015 yang berhasil dikumpulkan dan dipelajari sejumlah 70 paper dari berbagai publikasi. Metode pencarian adalah menggunakan fasilitas pencarian pada IEEE Explore, ACM, Google Schoolar. Pencarian juga menggunakan panduan dari repository yang disediakan pada crestweb.cs.ucl.ac.uk yang menyediakan daftar judul Paper mengenai topik SBSE, website tersebut juga digunakan sebagai referensi dalam menentukan kategorisasi dari paper saat terdapat kesulitan dalam menentukan kategori dari paper.

Dari fase requirement/specification, permasalahan yang paling sering dibahas masih sama dengan hasil yang didapatkan oleh survey dalam bidang ini di [8] yaitu permasalahan NRP (Next Release Problem), tetapi belum mendapati yang disebut sebagai permasalahan yang berkembang yaitu MONRP (Multi-Objective Next Release Problem).

Sama dengan review dari [5] dan [9] bahwa area yang paling banyak dibahas adalah testing, dan metode yang paling banyak digunakan adalah metode GA (Genetic Algorithm), sedangkan yang paling sedikit adalah pada bidang coding.

Arah penelitian yang disebutkan oleh Hamann [5] sudah mulai dieksplorasi, terutama pada arah topik ***multiobjective-optimization***, misalnya pada **testing** [10]–[13];[14], **disain** [15]–[17], **distribusi/ maintenance/ enhancement** [18]–[20], **verification** [21], dan **manajemen** [22]. Selain itu arah-arah penelitian yang sudah mulai dieksplorasi adalah : (1)

optimisasi yang menerapkan interaktifitas terhadap pengguna terutama untuk permasalahan dengan fungsi fitness yang sulit didefinisikan [17], [23], [24];[25], (2) **pengoptimisasian terhadap properti non-fungsional** dari system [19], [20], (3) **coevolusi**, yaitu optimasi menggunakan lebih dari 2 atau lebih populasi kandidat solusi yang bisa saling mempengaruhi antar populasi [19] [26] , (4) **on-line optimization**, yaitu penerapan pada permasalahan yang terus berubah termasuk pada saat optimasi dilakukan [27], dan (5) ***hybrid-optimization***, yaitu menggunakan kombinasi antara metode-metode metaheuristik seperti pada [28]–[30]; [31].

Selain arah penelitian diatas, pada [32] dilakukan eksplorasi terhadap potensi pengembangan baru yang disebut *incremental-optimisation*, yaitu optimasi yang dilakukan secara terbatas dan bertahap, untuk menjaga agar perubahan yang dilakukan tidak merusak struktur pengetahuan yang telah dibangun dalam organisasi atau dalam pikiran pengembang, melainkan perubahan dilakukan secara progresif dan bukan secara radikal seperti pada solusi optimasi dari proses yang tidak dibatasi. Walaupun hasil sementara yang didapatkan adalah bahwa *incremental-optimization* ini masih memiliki kekurangan dibandingkan pendekatan non-incremental yaitu dalam segi waktu optimasi, dan terdapatnya langkah 'persinggahan' yang nantinya tidak akan digunakan kemudian (dibuang), dan caveat terhadap ukuran langkah dengan ukuran instans, tetapi pendekatan ini sangat menjanjikan menjadi sebagai jalan untuk meningkatkan penggunaan SBSE pada dunia riil.

Faktor lain yang juga perlu diperhatikan agar memudahkan penggunaan metode SBSE menurut [33] adalah dengan menambahkan kemampuan adaptif pada proses pencarian, hal ini karena metode search-based memerlukan pemilihan parameter yang optimal untuk meningkatkan hasil yang didapatkan. Dan karena pemilihan parameter ini sangat mempengaruhi [34], maka pengguna SBSE yang kurang memiliki pengetahuan tentang SBSE akan memiliki kesulitan dalam memilih parameter yang tepat. Dengan kemampuan adaptif, metode pencarian yang digunakan akan secara otomatis mencari dan mengatur pencarian kepada nilai parameter yang lebih optimal, secara simultan dengan proses pencariannya sendiri. Hal ini membuat metode SBSE lebih mudah digunakan tanpa memerlukan pengetahuan khusus tentang metode optimasi yang digunakan.

Future Direction

Dari hasil survey yang dilakukan didapatkan beberapa selain arah yang telah diberikan pada survey-survey sebelumnya, beberapa tambahan arah yang menarik untuk dieksplorasi adalah:

1. SBSE yang mudah diterima.

Untuk lebih meningkatkan penggunaan SBSE, dibutuhkan agar penerapannya memperhatikan aspek kemudahan dalam penggunaan oleh organisasi atau domain spesialis atau pengembang (developer). Pendekatan yang sudah meiliki arah ini adalah dengan menggunakan *incremental-optimization* yang membuat perubahan dapat dilakukan secara bertahap dan dapat diikuti oleh pikiran pengembang, dan bukan perubahan secara radikal yang sering ditawarkan oleh optimasi SBSE. Pendekatan lain juga adalah tersedianya pilihan solusi yang didapatkan dari Multi-objective, dengan adanya pilihan beberapa solusi yang optimal dengan tingkat pemenuhan objetif yang berbeda-beda, pengguna dapat menggunakan pengetahuan subjetifnya (pengalaman, kriteria tacit, dll) untuk memilih solusi yang paling disukainya.

2. SBSE yang lebih tinggi dan luas

Untuk meningkatkan hasil yang didapatkan dari metode SBSE sebenarnya terdapat banyak ide yang dapat dikembangkan lebih lanjut dan sudah disebutkan dalam survey-survey terdahulu seperti hybrid-optimization coevolusi, dll. Penelitian-penelitian SBSE juga melakukan modifikasi dalam metode ataupun cara penerapan metode optimasi sehingga meningkatkan penerapannya, dan terlebih lagi penelitian dalam bidang spesifik terkait metode metaheuristik juga akan terus menghasilkan improvisasi atau bahkan mungkin metode metaheuristik yang lebih baru.

Tetapi yang paling menarik adalah penerapan penggunaan metode hyperheuristik, karena potensi yang dimilikinya untuk bekerja pada lapisan yang lebih tinggi, yaitu semacam metode-untuk-memilih-metode yang digunakan. Atau bahkan lebih menarik lagi adalah metode-untuk-membuat-metode dengan menggunakan kemampuan Genetic Programming untuk mengevolusi program atau metode pemecahan. Hal ini memungkinkan hyperheuristik mendapatkan hasil yang lebih tingg lagi (dengan memilih metaheuristik yang paling cocok untuk permasalahan) dan juga dapat digunakan secara luas (dengan kemampuan untuk membangkitkan metode baik dari nol maupun dari building block yang disediakan).

Kekurangan Dan Threat to Validity

Beberapa paper sebenarnya dapat dimsukkan pada beberapa kategori sekaligus misalnya misalnya misalnya (Nie dan Leung 2011) [40] yang masih pada kategori survey/review tetapi juga masuk pada kategori testing sehingga dalam paper ini terpaksa harus memilih salah satu yang dianggap paling kuat. Hal membuat kategorisasi yang dilakukan mungkin perlu diperbaiki untuk mendapatkan gambaran yang lebih jelas untuk mendapatkan peta penelitian yang lebih jelas. Jumlah paper yang menonjol pada tahun 2013 kemungkinan disebabkan pemilihan sumber paper yang terlalu terpusat pada satu atau lebih sumber. Jumlah paper yang diangkat masih terlalu sedikit untuk bisa menggambarkan tren secara keseluruhan (Harman and Mansouri 2010 [41] tahun 2008 saja sudah sampai 518 paper).

Survey belum memasukkan beberapa paper tentang pengembangan dari metode optimasi, karena alasan bahwa paper-paper ini tidak termasuk dalam pemecahan permasalahan Software Engineering. Padahal paper-

paper ini memiliki ide-ide yang sangat baik dan memiliki potensial untuk meningkatkan hasil yang didapatkan dalam penerapan metode search-based pada software engineering. Contoh paper seperti ini adalah [He et al. 2014] yang menggunakan pendekatan evaluasi fitness yang meningkatkan metode NSGA-II dan SPEA2 dalam memecahkan masalah dengan banyak objektif (lebih dari 2 dan 3 objektif), contoh lain adalah [Russo and Francisco 2014] yang membuat metode quick-hypervolume secara teoritis dan eksperimental meningkatkan metode-metode pencarian multi-objective. Pencarian paper masih menggunakan keyword umum yaitu "search-based software engineering", sehingga masih banyak kemungkinan melewatkan paper yang mungkin lebih spesifik ke suatu masalah software-engineering tertentu. Akan lebih baik kalau pencarian menggunakan keyword permasalahan yang spesifik, atau metode yang spesifik, juga metode yang bisa digunakan adalah melakukan penelusuran daftar pustaka dari paper-paper terlebih paper yang bersifat survey atau review dari SBSE atau sub-bidangnya (mil. Search Based Software Testing).

Paper dikategorisasi menggunakan pengetahuan pribadi sehingga masih besar kemungkinan terdapat bias dan kesalahan. Langkah kategorisasi yang dilakukan pertama berdasarkan kategori formal yang disebutkan dalam isi paper (mis. Subject descriptor/category), dan apabila tidak disebutkan secara eksplisit maka penulis menggunakan pengetahuan pribadinya berdasarkan pemahaman yang didapatkan dalam membaca paper tersebut. Untuk meyakinkan dalam melakukan kategorisasi, penentuan juga melihat kategorisasi dari sumber lain seperti dari repository crestweb (http://crestweb.cs.ucl.ac.uk).

3. Kesimpulan

Kelebihan metode/strategi metaheuristik dari metode heuristic adalah bahwa metaheuristik tidak problem spesifik sehingga dapat diaplikasikan pada banyak permasalahan, termasik pada permasalahan optimasi rekayasa perangkat lunak (SBSE). Metaheuristik terdiri dari beberapa jenis dengan kelebihan dan kekurangan masing-masing.

Penggunaan SBSE sangat cocok untuk diterapkan pada permasalahan optimasi RPL, tetapi tetap harus memperhatikan beberapa kendala yang bisa terjadi, termasuk didalamnya adalah dalam pemilihan metric untuk dijadikan fungsi fitness.

Paper topik SBSE tahun 2010-2015 walaupun belum secara lengkap dikumpulkan masih menggambarkan tren penelitian dibidang SBSE yang telah digambarkan sebelumnya pada tahun 2012 [5] yaitu: 1) tren peningkatan jumlah paper mengenai SBSE dari tahun ke tahun, 2) penerapan SBSE paling banyak adalah pada fase *software testing*, dan 3) metode yang paling banyak digunakan adalah *Genetic Algorithm*.

Topik yang disebut sebagai potensi penelitian lebih lanjut dari SBSE [5] sudah mulai dieksplorasi misalnya seperti Multi-Objective Optimization, Interactive Optimization, Non-functional Optimization, Online Optimization, dan Hybrid-Optimization. Selain arah penelitian diatas terdapat arah penelitian yang baik untuk dieksplorasi, yaitu *incremental optimization* untuk memudahkan solusi dari SBSE diterima pada industri/organisasi/individu pengguna dan adaptif-'SBSE yang membuat metode ini dapat digunakan oleh sebanyak mungkin individu karena penggunaannya tidak membutuhkan pengetahuan yang khusus dan juga menjamin mendapatkan hasil yang paling optimal karena menggunakan parameter terbaik yang dipilih secara otomatis.

Future Works

Pengembangan lebih lanjut adalah memperluas survey yang dilakukan dengan meningkatkan jumlah paper paper yang disurvey. Penggalian lebih mendalam tentang permasalahan spesifik yang menjadi tren dalam setiap fase software engineering. Melihat secara lebih detil jumlah objective dari optimasi yang dilakukan. Analisis yang lebih mendalam juga diperlukan mengenai kendala-kendala dalam penggunaan metode SBSE, sehingga dapat menjadi bekal peneliti baru dalam melakukan eksplorasi terhadap bidang ini.

Daftar Pustaka

[1] M. Harman, "The Current State and Future of Search Based Software Engineering The Current State and Future of Search Based Software Engineering," 2007.

[2] M. Harman dan J. Clark, "Metrics are fitness functions too," in *Proceedings - International Software Metrics Symposium*, 2004, pp. 58–69.

[3] L. Bianchi, M. Dorigo, L. M. Gambardella, dan W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Nat. Comput.*, vol. 8, no. 2, pp. 239–287, 2009.

[4] C. Blum dan A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[5] M. Harman, S. A. Mansouri, dan Y. Zhang, "Search-based software engineering: Trends, Technique, and Application," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 1–61, Nov. 2012.

[6] M. Harman, P. McMinn, J. De Souza, dan S. Yoo, "Search based software engineering: Techniques, taxonomy, tutorial," *Empir. Softw. Eng. Verif.*, pp. 1–59, 2012.

[7] M. Harman, "The relationship between search based software engineering and predictive modeling," *Proc. 6th Int. Conf. Predict. Model. Softw. Eng. - PROMISE '10*, p. 1, 2010.

[8] A. M. Pitangueira, R. S. P. Maciel, dan M. D. O. Barros, "A Systematic Review of Software Requirements Selection and Prioritization Using SBSE Approaches," pp. 188–208, 2013.

[9] S. R. Vergilio, T. E. Colanzi, A. T. R. Pozo, dan W. K. G. Assuncao, "Search Based Software Engineering: A Review from the Brazilian Symposium on Software Engineering," *2011 25th Brazilian Symp. Softw. Eng.*, pp. 50–55, Sep. 2011.

[10] S. Wang, S. Ali, dan A. Gotlieb, "Minimizing test suites in software product lines using weight-based genetic algorithms," *Proceeding fifteenth Annu. Conf. Genet. Evol. Comput. Conf. - GECCO '13*, p. 1493, 2013.

[11] G. Assun, T. E. Colanzi, S. R. Vergilio, dan A. Pozo, "On the Application of the Multi-Evolutionary and Coupling-Based Approach with Different Aspect-Class Integration Testing Strategies," pp. 19–33, 2013.

[12] J. Shelburg, M. Kessentini, dan D. R. Tauritz, "Regression Testing for Model Transformations : A Multi-objective Approach," no. 1, pp. 209–223, 2013.

[13] A. Panichella, R. Oliveto, M. Di Penta, dan A. De Lucia, "Improving Multi-Objective Test Case Selection by Injecting

Diversity in Genetic Algorithms," *IEEE Trans. Softw. Eng.*, vol. 5589, no. c, pp. 1–1, 2014.

[14] L. Briand, Y. Labiche, dan K. Chen, "A multi-objective genetic algorithm to rank state-based test cases," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 8084 LNCS, pp. 66–80.

[15] A. Ramírez, J. R. Romero, dan S. Ventura, "On the Performance of Multiple Objective Evolutionary Algorithms for Software Architecture Discovery," pp. 1287–1294, 2014.

[16] M. Bowman, L. C. Briand, dan Y. Labiche, "Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 817–837, 2010.

[17] C. L. Simons, I. C. Parmee, dan R. Gwynllyw, "Interactive, evolutionary search in upstream Object-oriented class design," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 798–816, 2010.

[18] M. . S. H. . B. S. . O. a Kessentini W.; Kessentini, W. Kessentini, M. Kessentini, H. Sahraoui, S. Bechikh, dan A. Ouni, "A Cooperative Parallel Search-Based Software Engineering Approach for Code-Smells Detection," *IEEE Trans. Softw. Eng.*, vol. 40, no. 9, pp. 841–861, Sep. 2014.

[19] and J. A. C. David R. White, Andrea Arcuri, "Evolutionary Improvement of Programs," vol. 15, no. 4, pp. 515–538, 2013.

[20] W. B. Langdon dan M. Harman, "Optimising Existing Software with Genetic Programming," *IEEE Trans. Evol. Comput.*, vol. PP, no. 1, pp. 1–18, 2014.

[21] S. Poulding dan J. a. Clark, "Efficient software verification: Statistical testing using automated search," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 763–777, 2010.

[22] A. S. Sayyad, T. Menzies, dan H. Ammar, "On the value of user preferences in search-based software engineering: A case study in software product lines," *2013 35th Int. Conf. Softw. Eng.*, pp. 492–501, May 2013.

[23] L. Ben Said, S. Bechikh, dan K. Ghedira, "The r-Dominance: A new dominance relation for interactive evolutionary multicriteria decision making," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 801–818, 2010.

[24] B. Marculescu, R. Feldt, dan R. Torkar, "Objective Re-weighting to Guide an Interactive Search Based Software Testing System," *2013 12th Int. Conf. Mach. Learn. Appl.*, pp. 102–107, Dec. 2013.

[25] C. L. Simons dan I. C. Parmee, "Elegant object-oriented software design via interactive, evolutionary computation," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1797–1805, 2012.

[26] S. Nguyen, M. Zhang, M. Johnston, dan K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 193–208, 2014.

[27] W.-N. Chen dan J. Zhang, "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler," *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, pp. 1–17, 2012.

[28] J. Smith dan C. Simons, "A Comparison of Two Memetic Algorithms for Software," pp. 1485–1492, 2013.

[29] M. Harman dan P. McMinn, "A theoretical and empirical study of search-based testing: Local, global, and hybrid search," *IEEE Trans. Softw. Eng.*, vol. 36, no. 2, pp. 226–247, 2010.

[30] P. McMinn, M. Harman, K. Lakhotia, Y. Hassoun, dan J. Wegener, "Input domain reduction through irrelevant variable removal and its effect on local, global, and hybrid search-based structural test data generation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 453–477, 2012.

[31] Z. Wang, X. Yao, dan K. Tang, "A memetic algorithm for Multi-Level Redundancy Allocation," *IEEE Trans. Reliab. VOL. 59, NO. 4, DECEMBER 2010*, vol. 203, no. 4, pp. 241–250, 2010.

[32] M. De Oliveira Barros, "An experimental study on incremental search-based software engineering," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 8084 LNCS, pp. 34–49.

[33] M. Harman, E. Burke, J. a Clark, dan X. Yao, "Dynamic Adaptive Search Based Software Engineering," in *Proceedings of the 6th International Symposium on Empirical Software Engineering and Measurement (ESEM '12) (Keynote)*, 2012, pp. 1–8.

[34] A. S. Sayyad, K. Goseva-Popstojanova, T. Menzies, dan H. Ammar, "On Parameter Tuning in Search Based Software Engineering: A Replicated Empirical Study," *2013 3rd Int. Work. Replication Empir. Softw. Eng. Res.*, pp. 84–90, Oct. 2013.

[35] M. Harman, S. A. Mansouri, dan Y. Zhang, "Search Based Software Engineering : A Comprehensive Analysis and Review of Trends Techniques and Applications," pp. 1–78, 2009.

[36] E. Zitzler, M. Laumanns, dan S. Bleuler, "A Tutorial on Evolutionary Multiobjective Optimization," in *Metaheuristics for Multiobjective Optimisation*, 2004, pp. 3–37.

[37] K. Deb dan K. Deb, "Current trends in evolutionary multi-objective optimization," *Int. J. Simul. Multidiscip. Des. Optim.*, vol. 1, pp. 1–8, 2007.

[38] V. Khare, X. Yao, dan B. Sendhoff, "Credit assignment among neurons in co-evolving populations," in *Parallel Problem Solving from Nature (PPSN)*, 2004, pp. 882–891.

[39] D. E. Moriarty dan M. Rey, "Forming Neural Networks through Efficient and Adaptive Coevolution," 1998.

[40] C. Nie dan H. Leung, "A survey of combinatorial testing," *ACM Comput. Surv.*, vol. 43, no. 2, pp. 1–29, 2011.

[41] M. Harman dan A. Mansouri, "Search based software engineering: Introduction to the special issue of the IEEE transactions on software engineering," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 737–741, Nov. 2010.

Biodata Penulis

Mohamad Syafri Tuloli, memperoleh gelar Sarjana Teknik (ST), Jurusan Teknik Informatika Universitas Islam Indonesia Yogyakarta, lulus tahun 2005. Memperoleh gelar Magister Teknik (MT) Program Pasca Sarjana Magister Teknik Informatika Institut Teknologi Bandung, lulus tahun 2007. Bekerja sebagai Dosen di Universitas Negeri Gorontalo, dan saat ini sedang mengikuti studi program doktorat di Institut Teknologi Bandung.

Prof. Dr. Ing. Benhard Sitohang, adalah Guru Besar di Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. Ketua kelompok keahlian Rekayasa Perangkat Lunak dan Data. Menjadi promotor untuk mahasiswa program doktorat dalam bidang terkait Rekayasa Perangkat Lunak dan Data. Salah satu arah riset adalah terkait sistem basis data, misalnya untuk aspek skalabilitas dan performansi dari manajemen data yang sedang berkembang.

Dr. Bayu Hendradjaya, Saat ini menjadi Dosen di Sekolah Teknik Elektro dan Informatika, mengajarkan Rekayasa Perangkat Lunak, Pemrograman, Kualitas Perangkat Lunak, dan Keamanan dalam Pengembangkan Perangkat Lunak. Memiliki pengalaman praktis yang luas dalam pengembangan perangkat lunak. Menjadi pembimbing untuk mahasiswa program doktorat dalam bidang terkait Rekayasa Perangkat Lunak.