khazanah informatika

Jurnal Ilmu Komputer dan Informatika

Hyper-heuristics untuk Penyelesaian Masalah Optimasi Lintas Domain dengan Seleksi Heuristik berdasarkan *Variable Neighborhood Search*

Nisa Dwi Angresti*, Arif Djunaidy, Ahmad Muklason
Departemen Sistem Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
*nisa.dwi.angresti@gmail.com

Abstrak-Metode *state-of-the-art* untuk menyelesaikan permasalahan optimasi kombinatorik yang diketahui sebagai permasalahan *non-deterministic polynomial hard* (NP-hard) adalah *meta-heuristics*. Kelemahan dari metode *meta-heuristics* adalah dibutuhkanya *parameter tuning* yang spesifik untuk setiap *problem domain* berbeda. Hal ini menyebabkan pendekatan *meta-heuristics* kurang efektif untuk menyelesaikan permasalahan lintas *problem domain*. Untuk mengatasi permasalahan tersebut, muncul pendekatan baru, yaitu *hyper-heuristics*. Dengan meningkatkan level *search space* dari *solution space* ke *low-level heuristics* space, *hyper-heuristics* diharapkan dapat menjadi pendekatan yang lebih general dan efektif untuk menyelesaikan permasalahan lintas problem domain. Penelitian ini bertujuan untuk menginvestigasi performa algoritma *variable neighborhood search* (VNS) sebagai strategi untuk memilih *low-level heuristics* dalam kerangka kerja *hyper-heuristics*. Hal ini berbeda dengan penelitian-penelitian sebelumnya di mana VNS digunakan dalam kerangka kerja *meta-heuristics*. Metode yang diusulkan dalam penelitian ini diuji coba pada 6 problem domain yang berbeda, yaitu *satisfiability* (SAT), *one dimensional bin packing, permutation flow shop, personel scheduling, travelling salesman problem* (TSP), dan *vehicle routing problem* (VRP). Hasil komputasi menunjukkan bahwa metode VNS ini memiliki performa lebih baik dibandingkan dengan metode seleksi *low-level heuristics* pembanding, yaitu *Simple Random*. Secara lebih spesifik, VNS lebih unggul pada 5 dari 6 problem domain.

Kata kunci: Optimasi Lintas Domain; Hyper-heuristics; Variable Neighborhood Search

1. Pendahuluan

Permasalahan optimasi kombinatorik adalah proses mencari satu solusi yang paling optimal dari kumpulan solusi yang tersedia [1]. Model optimasi kombinatorik berfungsi untuk memodelkan berbagai bentuk masalah dalam beberapa fungsi tujuan yang menunjukkan kualitas solusi yang diberikan, kemudian menggunakan algoritma pencarian tertentu untuk meminimalkan atau memaksimalkan fungsi tujuan tersebut [2]. Optimasi penting digunakan untuk membantu pengambilan keputusan dalam berbagai bidang, seperti bidang industri, pemerintahan, pendidikan, dan transportasi.

Permasalahan optimasi saat ini berkembang menjadi permasalahan yang kompleks karena banyaknya jumlah *input* dataset dan batasan yang harus dipenuhi dalam menentukan optimalitas. Beragam bentuk masalah optimasi kombinatorik berkembang pada berbagai bidang, di antaranya adalah masalah mengoptimalkan pengemasan barang, penjadwalan mesin, penjadwalan mata kuliah, dan penentuan rute kendaraan. Sebuah organisasi dapat membutuhkan banyak optimasi untuk mendukung bisnisnya. Sebagai contoh perusahaan yang bergerak dalam bidang logistik, tidak hanya membutuhkan optimasi untuk satu bidang, tetapi membutuhkan optimasi untuk banyak bidang, seperti penjadwalan sopir, rute kendaraan, dan pengemasan barang (*bin packing*) sekaligus.

Dalam mencari solusi optimal dibutuhkan berbagai jenis metode atau algoritma pencarian. Semakin banyaknya masalah optimasi, maka semakin banyak metode pencarian yang dikembangkan untuk menyelesaikan masalah tersebut. Secara garis besar, metode atau algoritma pencarian dalam menyelesaikan masalah optimasi dapat dikelompokkan menjadi dua kelompok, yaitu algoritma *exact* dan algoritma *approximate* [3] Algoritma *exact* digunakan untuk menyelesaikan masalah optimasi yang sederhana. Dalam menyelesaikan masalah sederhana, algoritma *exact* dapat menemukan solusi yang paling optimal dengan waktu yang cepat. Namun, pada masalah optimasi yang kompleks seperti permasalahan *Traveling Salesman Problem* (TSP), *Vehicle Routing Problem* (VRP), dan penjadwalan dengan *input* dataset yang besar, algoritma *exact* belum mampu mencari solusi yang paling optimal dalam *running time* yang cepat. Oleh karena itu, algoritma *approximate* seperti *heuristics*, *meta-heuristics*, dan *hyper-heuristik* digunakan sebagai pilihan dalam menyelesaikan masalah optimasi yang kompleks. Algoritma *approximate* lebih mengutamakan pencarian yang cepat daripada pencarian yang lengkap, yakni *exhaustive search* sehingga efisien. Namun, solusi yang ditemukan dengan metode tersebut bukanlah solusi yang paling optimal, tetapi cukup baik dan dapat diselesaikan dengan waktu yang singkat, yakni *polynomial time*.

Pengembangan metode *heuristics* adalah *meta-heuristics* yang menyelesaikan masalah optimasi secara spesifik berdasarkan karakteristik satu permasalahan tertentu [4]. Untuk menyelesaikan banyak masalah optimasi, *meta-heuristics* harus mendeskripsikan dan memodelkan setiap permasalahan, serta menentukan metode pencarian yang tepat untuk menyelesaikan setiap masalah. Penyelesaian masalah optimasi secara spesifik membutuhkan biaya yang mahal dan waktu yang lama, karena jika ada masalah optimasi baru, metode tersebut harus memodelkan masalah dan memodifikasi metode pencarian dari awal serta diperlukan penentuan *parameter tuning* untuk setiap masalah. Penerapan metode tersebut dapat memperlambat proses pengambilan keputusan untuk mengoptimasikan banyak permasalahan yang berbeda (lintas domain). Oleh karena itu, perlu suatu metode yang dapat menyelesaikan berbagai masalah optimasi secara general sehingga dapat mempercepat pencarian solusi [5]

Hyper-heuristics adalah metodologi otomatis untuk mencari atau membuat heuristic baru dalam meningkatkan generalitas permasalahan dan *instance dataset* berbagai permasalahan optimasi yang kompleks [4]. Dalam menyelesaikan berbagai permasalahan optimasi, *hyper-heuristics* memiliki dua tingkatan proses, yaitu *high level heuristic* dan *low level heuristic* (LLH). *Hyper-heuristics* merupakan *high level heuristic* yang melakukan manipulasi terhadap LLH yang terdapat pada setiap masalah spesifik. LLH merupakan *heuristics* yang melakukan pencarian solusi pada domain masalah optimasi. Proses pencarian yang bertingkat ini menjadi kerangka *hyper-heuristics* dalam meningkatkan generalitas permasalahan optimasi.

Walaupun *hyper-heuristics* dapat meningkatkan generalitas pencarian pada banyak permasalahan optimasi, tetapi kinerja *hyper-heuristics* dalam menemukan solusi optimal tidak selalu baik untuk semua permasalahan optimasi [5]. Hal tersebut disebabkan oleh perbedaan karakteristik permasalahan optimasi. Perbedaan karakteristik ini menyebabkan LLH pada setiap permasalahan optimasi juga berbeda-beda. Kinerja *hyper-heuristics* masih harus ditingkatkan untuk menciptakan hasil yang lebih optimal. *High level heuristic* harus mengenali dan memanfaatkan LLH yang berbeda pada setiap permasalahan optimasi agar dapat menghasilkan solusi yang baik lebih optimal. *High level heuristic* memiliki dua mekanisme utama, yaitu mekanisme seleksi LLH dan *move acceptance*. Mekanisme seleksi LLH berfungsi untuk menemukan LLH yang diaplikasikan dalam menghasilkan solusi baru, sedangkan mekanisme *move acceptance* berfungsi untuk memutuskan apakah akan menerima solusi yang dihasilkan oleh LLH tersebut atau tidak [6]. Pencarian yang dilakukan di dalam proses tersebut harus menemukan optimalitas secara global. Mekanisme seleksi LLH dan *move acceptance* pada strategi seleksi *high level heuristic* ini dilakukan secara terpisah. Kombinasi dari kedua mekanisme ini menjadi strategi *high level heuristic* untuk meningkatkan kinerja dalam menemukan solusi optimal pada berbagai permasalahan optimasi.

Penerapan *hyper-heuristics* telah dilakukan pada penelitian sebelumnya. Dalam penelitian [7] diusulkan strategi *high level heuristic* menggunakan metode *simple random* yang dikombinasikan dengan metode *adaptive iteration limited list-based threshold Acceptance*. Penelitian [8] mengusulkan strategi *high level heuristic* menggunakan metode *simple random* yang dikombinasikan dengan metode *late acceptance*.

Penelitian terdahalu tersebut telah berhasil menerapkan strategi *high level heuristic* untuk menyelesaikan beberapa permasalahan optimasi dengan mekanisme seleksi LLH menggunakan metode *simple random*. Namun, metode *simple random* kurang efektif karena pencarian LLH dilakukan secara acak. Seleksi LLH tidak merata. LLH yang terpilih secara acak belum tentu optimal, bisa jadi LLH yang tidak terpilih adalah LLH yang optimal. Oleh karena itu, perlu adanya peningkatan kinerja dari *high level heuristic* dalam menemukan solusi yang optimal pada berbagai permasalahan optimasi.

Penelitian ini mengusulkan strategi *high level heuristic* yang baru dalam seleksi LLH dan *move acceptance* agar dapat meningkatkan kinerja pencarian setiap permasalahan optimasi. Dalam penelitian ini, seleksi LLH dilakukan dengan mengadaptasi pendekatan *variable neighborhood search (VNS)*. VNS merupakan teknik pencarian solusi yang mengubah lingkungan secara sistematis dalam pencarian untuk menghasilkan *global optimal* [9]. VNS melibatkan pencarian lokal secara berurutan untuk mengubah solusi saat sehingga menjadi optimal secara lokal. Prosedur ini efektif untuk meningkatkan intensifikasi yang memakan waktu [10]. VNS efisien dalam meningkatkan hasil pencarian [11].

Selain seleksi LLH, komponen penting lainnya yang menentukan keberhasilan *hyper-heuristics* adalah mekanisme *move acceptance*. Dalam penelitian ini, VNS dikombinasikan dengan mekanisme *move acceptance* Hill Climbing. Hasil yang diharapkan dari penelitian ini adalah strategi *high level heuristic* yang dapat meningkatkan kinerja *hyper-heuristics* dalam menemukan solusi yang lebih optimal pada masalah optimasi lintas domain.

Strategi *high level heuristic* sebagai seleksi *hyper-heuristics* yang diusulkan, diuji coba menggunakan *framework Hyper-Heuristics Flexible* (HyFlex)[1]. HyFlex merupakan sebuah *framework* yang menyediakan antarmuka yang mudah digunakan untuk pengembangan dan metolodogi pencarian seleksi *hyper-heuristics* dengan implementasi beberapa domain masalah optimasi berbeda yang telah memiliki representasi solusi dan *low level heuristic* [12]. *Problem domain* pada HyFlex ini merupakan *benchmark* dari permasalahan optimasi kombinatorik, di mana kinerja teknik adaptasi lintas domain yang berbeda dapat dinilai dan dibandingkan. Terdapat enam *problem domain* masalah optimasi kombinatorial pada *framework* HyFlex, yaitu *satisfiability* (SAT), *one dimensional bin packing*, *permutation flow shop*, *personnel scheduling*, TSP, dan VRP [13]. Hasil implementasi strategi *high level heuristic* pada HyFlex akan dibandingkan dengan strategi

[1]http://intip.in/hyperheuristics

high level heuristic pada penelitian sebelumnya yang diimplementasikan ulang.

2. Kajian Pustaka

Penelitian *hyper-heuristics* terdahulu dilakukan dengan mengkombinasikan metode untuk mekanisme seleksi LLH dan *move acceptance*. Penelitian [8] telah mengusulkan strategi *high level heuristic* dengan menguji efek kinerja mekanisme pemilihan stokastik sederhana, i.e. *simple random,* terhadap pemilihan rangkaian LLH di dalam kerangka *hyper-heuristics* dengan memperkenalkan kelas baru metode pemilihan heuristik berdasarkan pilihan *roulette-wheel* dan menggabungkannya dengan metode *Late Acceptance. Simple Random* digunakan untuk menerapkan LLH secara acak dari LLH yang tersedia. Penelitian tersebut diuji coba terhadap enam *problem dom ain* pada HyFlex. Penelitian *hyper-heuristics* tentang mencari nilai *parameter tuning* telah dibahas dalam penelitian [14].

Penelitian [15] mengusulkan strategi *high level heuristic* yang dikenal sebagai *gene expression programming hyperheuristics (GEP-HH)*. Pada model GEP-HH, mekanisme seleksi LLH menggunakan pendekatan *dynamic multi-armed bandit-extreme value based reward* dan metode *move acceptance* menggunakan *gene expression programming (GEP)*. Model tersebut telah menerapkan seleksi *hyper-heuristics* untuk menyelesaikan permasalahan kombinatorik lintas domain, yaitu: *exam timetabling* dan *dynamic vehicle routing problem (DVRP)*. Pada *exam timetabling*, waktu eksekusi ditentukan dengan menggunakan patokan *software* untuk memastikan keadilan (*fair*) perbandingan antara penelitian yang dilakukan dan *platform* lainnya. Waktu eksekusi yang ditentukan adalah 10 menit. Pada *dynamic vehicle routing*, waktu eksekusi disesuaikan seperti penelitian sebelumnya, yaitu 750 detik. Hasil penerapan model GEP-HH dibandingkan dengan metode *state-of-the-art* untuk DVRP. Hasil komputasional menunjukkan bahwa secara umum, model GEP-HH lebih unggul dari metode yang telah ada sebelumnya. Pada penelitiam [11], VNS digunakan dalam *framework meta-heuristics* dan hanya digunakan dalam satu *problem domain* saja, yaitu VRP.

Metode *hyper-heuristics* juga sudah banyak digunakan untuk menyelesaikan permasalahan optimasi kombinatorik lainya seperti pada [16] [17] yang digunakan untuk menyelesaikan permasalahan *multi-objective examination timetabling problem*. Sedangkan pada [18-19], *hyper-heuristics* digunakan untuk optimasi rencana perjalanan dengan menggunakan moda transportasi umum. Keterbaruan dari penelitian adalah bahwa ini adalah studi pertama yang menginvestigasi metode VNS sebagai strategi dalam *hyper-heuristics*, yang tidak hanya untuk menyelesaikan satu permasalahan yang spesifik, tetapi enam permasalahan lintas domain sekaligus.

3. Metode

a. Identifikasi Masalah

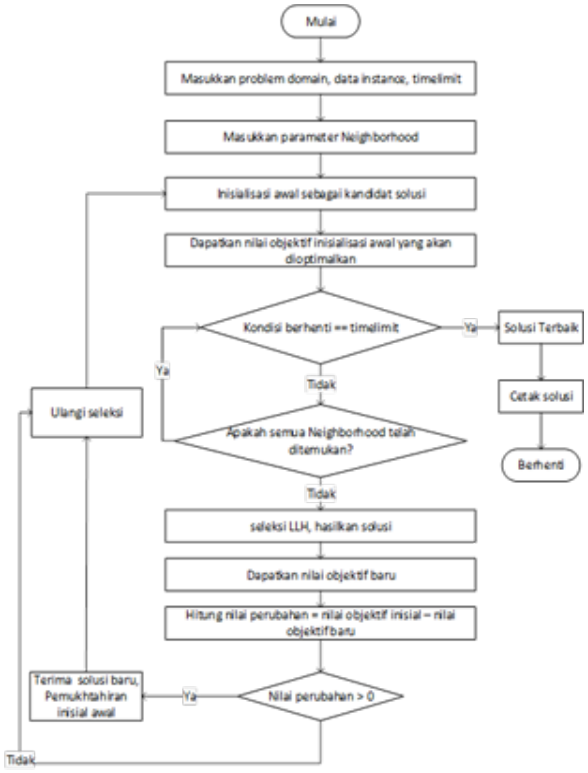
Penelitian ini mengembangkan metode *hyper-heuristics* sebagai strategi *high level heuristic* dalam memilih LLH dari rangkaian LLH yang telah ada pada masing-masing domain masalah optimasi. *Hyper-heuristics* adalah sebuah metode, algoritma, atau mekanisme pembelajaran untuk mencari atau menghasilkan heuristik sehingga dapat menyelesaikan masalah pencarian komputasional [5]. Tujuan dari penelitian ini adalah meningkatkan kinerja dari *hyper-heuristics* dalam mengoptimalkan masalah optimasi lintas domain.

b. Datasets

Data yang digunakan adalah data yang terdapat pada HyFlex. HyFlex dapat dilihat sebagai *package framework* dan *benchmark datasets* untuk optimasi kombinatorik, di mana kinerja teknik adaptasi lintas domain yang berbeda dapat dinilai dan dibandingkan dengan baik. Terdapat enam domain permasalahan kombinatorial yang secara penuh diimplementasikan pada HyFlex, yaitu *satisfiability, one dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman problem (TSP)*, dan *vehicle routing problem (VRP)*. *Satisfiability* merupakan permasalahan pemenuhan aljabar boolean. Masalah *bin packing* adalah masalah mengoptimalkan penggunaan sebuah bin dalam pengemasan barang. Masalah *flow shop* adalah masalah untuk mengoptimalkan urutan kerja mesin sehingga pekerjaan cepat selesai. Masalah *personnel schedulling* adalah masalah mengoptimalkan jam kerja karyawan sehingga tidak ada karyawan yang memiliki jam kerja lebih banyak. Permasalahan TSP adalah masalah mengoptimalkan jarak yang ditempuh oleh seseorang dalam mengunjungi beberapa tempat. Masalah VRP adalah masalah mengoptimalkan rute kendaraan yang harus dikunjungi oleh kendaraan. Setiap permasalahan optimasi menggunakan data yang nyata (*instance*) dari HyFlex. Untuk menggunakan data pada HyFlex, dapat memanggil fungsi-fungsi (*methods*) yang terdapat pada HyFlex.

c. Desain Algoritma

Desain *hyper-heuristics* ini merupakan pengembangan *hyper-heuristics* sebagai strategi *high level heuristic* yang diusulkan. Pada tahap ini, *high level heuristic* akan menyeleksi salah satu *low level heuristic* dari rangkaian *low level heuristic* yang ada. Pada setiap iterasi, *low level heuristic* yang terpilih akan menghasilkan sebuah solusi baru, kemudian solusi yang dihasilkan tersebut akan dipertimbangkan untuk diterima atau tidak untuk menggantikan solusi sebelumnya berdasarkan kriteria *move acceptance*. Jadi, ada dua strategi atau metode yang menentukan kinerja dari *hyper-heuristics*, yaitu: metode memilih LLH, dan metode *move acceptance*.



Gambar 1. *High Level* Strategi yang Diusulkan

Tabel 1. Rangkaian *Low Level Heuristic* setiap Domain

No	Permasalahan optimasi	*Mutation*	*Ruin-Recreate*	*Crossover*	*Local Search*	Total
1	*Satisfiability*	6	1	2	2	11
2	*One Dimensional Bin Packing*	3	2	1	2	8
3	*Permutation Flow Shop*	5	2	3	4	15
4	*Personnel scheduling*	1	3	3	4	12
5	*Traveling salesman Problem*	5	1	3	2	13
6	*Vehicle Routing Problem*	3	2	2	3	10

Dalam penelitian ini, metode seleksi LLH *Variable Neighborhood Search (VNS)* dikombinasikan dengan metode *move acceptance Hill Climbing* yang selanjutnya disebut VNS-HC. Di dalam strategi ini, setiap pencarian LLH dilakukan secara sistematis berdasarkan urutan *neighborhood*, dan setiap LLH yang terpilih akan membangkitkan satu solusi karena pencarian berbasis *single-point-search*. Untuk *move acceptance*, metode ini hanya menerima solusi yang mengalami peningkatan. Pada desain *high level heuristic* VNS-HC ini dimulai dengan inisialisasi solusi untuk mendapatkan solusi awal. Dari solusi awal ini didapatkan nilai fungsi tujuan yang akan menjadi pembanding awal dengan solusi baru yang akan dihasilkan pada iterasi berikutnya. Untuk melihat detail proses dari strategi *high level* ini dapat dilihat pada Gambar 1.

Low Level Heuristic (LLH) yang digunakan di dalam penelitian ini adalah heuristik yang terdapat pada setiap permasalahan optimasi di dalam *framework* HyFlex [13]. Rangkaian LLH dalam HyFlex dikelompokkan menjadi empat kelompok, yaitu mutasi, *ruin-recreate*, *local search*, dan *crossover*.

Kelompok heuristik mutasi melakukan perubahan secara acak pada solusi dengan cara menukar, mengubah, membuang, menambah, atau menghapus komponen solusi. Pada kelompok *ruin-recreate* perubahan pada solusi dilakukan dengan cara merombak sebagian solusi dan membangun atau membuatnya kembali. Kelompok heuristik *local search* secara iteratif membuat perubahan kecil pada solusi, dan hanya menerima solusi yang meningkat sampai kondisi *local optimum* ditemukan atau kondisi berhenti terpenuhi. Heuristik ini berbeda dari kelompok heuristik mutasi karena heuristik menggabungkan proses perbaikan berulang dan mereka menjamin bahwa solusi yang baik akan dihasilkan. Pada kelompok heuristik *crossover* solusi baru dihasilkan dengan cara melakukan perubahan dengan mengambil dua solusi, menggabungkannya dan mengembalikan solusi baru. Tabel 1 merupakan jumlah rangkaian low level heuristic yang terdapat pada HyFlex.

d. Implementasi

Pada tahap ini, strategi *high level heuristic* yang diusulkan dan strategi *high level heuristic* pada penelitian sebelumnya diimplementasikan. Implementasi dalam penelitian ini dilakukan pada komputer dengan prosesor Intel Pentium N3710 1.6 GHz dan memori 4096MB. Hasil desain algoritma diimplementasikan pada framework HyFLex [13] menggunakan *tools* NetBeans IDE 8.2. Implementasi pada HyFlex dilakukan dengan cara memanggil fungsi-fungsi (methods) dan *libraries* yang ada pada HyFlex.

e. Uji Coba

Tahap uji coba ini merupakan tahap untuk melihat kinerja strategi *high level* heuristic yang diusulkan dengan strategi *high level* heuristic pada penelitian sebelumnya. Uji coba dilakukan dengan cara menjalankan algoritma sebanyak 31 kali eksekusi pada setiap *instance* dari CheSC dengan *running time* (*elapsed time*) yang sama. Batasan *running time* dan jumlah 31 kali ini berdasarkan literatur sebelumnya pada [7] dan [8]. Uji coba ini menghasilkan nilai fungsi *fitness* (untuk dicari solusi paling minimal) dari masing-masing *problem instance*. Pada pengujian ini terdapat lima *instance* pada setiap enam *problem domain*, sehingga terdapat 30 variasi *instance* data. Setiap strategi *high level heuristic* diujicoba dengan data yang sama. Dari hasil eksekusi, didapatkan penyebaran data nilai fungsi *fitness* setiap *instance*. Dari data hasil eksekusi tersebut dihitung nilai terbaik (minimal), kuartil pertama, median, kuartil ketiga, nilai maksimal, dan rata-rata yang akan dijadikan nilai perbandingan untuk setiap metode. Untuk melihat hasil uji coba nilai fungsi *fitness* terhadap masing-masing strategi dapat dilihat pada Tabel 2 dan Tabel 3.

4. Hasil dan Diskusi

Untuk menguji kinerja VNS-HC, dilakukan perbandingan dengan strategi *high level heuristic Simple Random-Hill Climbing* (Rand-HC) dalam penelitian sebelumnya yang diimplementasikan ulang. Kinerja diuji berdasarkan analisis data dari nilai minimal, kuartil pertama, median, kuartil ketiga, dan nilai maksimal fungsi *fitness* yang dihasilkan pada saat eksekusi. Pengujian kinerja pada masing-masing strategi dilakukan dengan tiga perbandingan. Pertama, membandingkan penyebaran data secara keseluruhan dari hasil eksekusi masing-masing strategi menggunakan diagram *boxplot*. Kedua, secara spesifik membandingkan nilai median untuk mengukur pemusatan data yang menjadi nilai tengah dari sekelompok data. Ketiga, membandingkan nilai minimal fungsi *fitness* hasil eksekusi masing-masing strategi. Karena fungsi tujuan dari setiap permasalahan optimasi adalah meminimalkan, maka kinerja dari strategi *high level heuristic* yang diusulkan dalam menghasilkan nilai minimal fungsi objektif perlu diuji. Secara umum, hasil penelitian secara komputasi dengan algoritma *hyper-heuristic Random - Hill Climbing (Rand-HC)* dapat dilihat pada Tabel 2, sedangkan hasil dari algoritma *hyper-huristic Variable Neighbourhood Search - Hill Climbing (VNS-HC)* dapat dilihat pada Tabel 3.

Tabel 2. Hasil Eksekusi Rand-HC

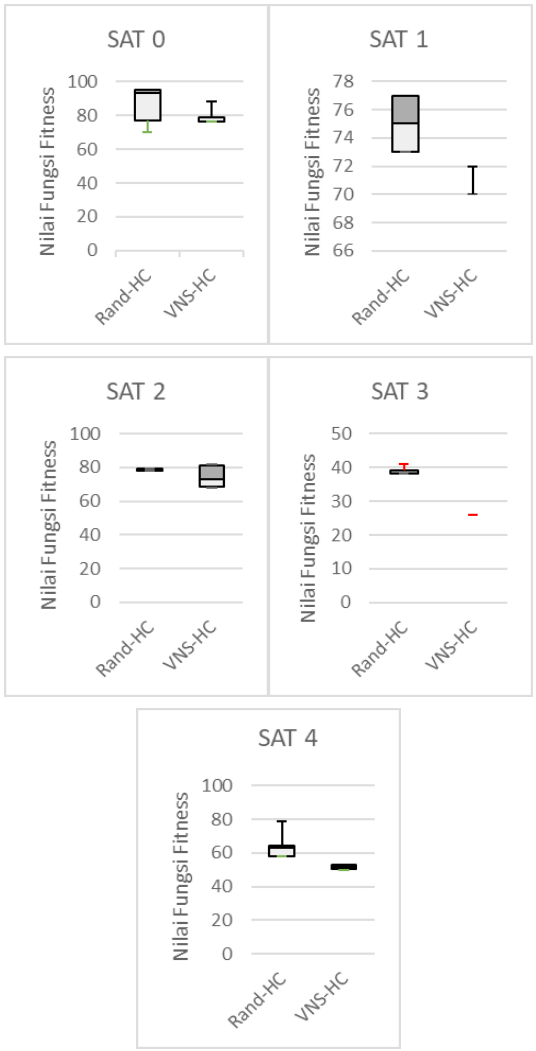
Permasalahan	Instance	Min	Q1	Median	Q3	Maksimal	Rata-Rata
SAT	0	70	77	93	95	95	86,45161
	1	73	73	75	77	77	75,19355
	2	78	78	78	79	79	78,35484
	3	38	38	38	39	41	38,70968
	4	58	58	63	64	79	62,22581
Bin Packing	0	0,101623	0,101623	0,101623	0,101623	0,101623	0,101623
	1	0,099989	0,099989	0,099989	0,099989	0,099989	0,099989
	2	0,107186	0,107186	0,107186	0,107186	0,107186	0,107186
	3	0,084335	0,084341	0,084341	0,084341	0,084341	0,084341
	4	0,04753	0,04753	0,04753	0,04753	0,04753	0,04753
Flow Shop	0	6424	6424	6497	6497	6534	6473,806
	1	6415	6415	6464	6464	6464	6448,194
	2	6507	6507	6507	6507	6507	6507
	3	6491	6491	6491	6491	6491	6491
	4	6517	6517	6517	6539,5	6540	6523,645
Personnel Scheduling	0	3406	3422,5	3446	3464,5	3512	3446,323
	1	2538	2870	3545	3907,5	5402	3498,903
	2	445	715	1870	2027,5	3475	1606,484
	3	33	43	46	49,5	337	68
	4	38	43	45	52	148	50,58065

Permasalahan	Instance	Min	Q1	Median	Q3	Maksimal	Rata-Rata
	0	51426,79	51426,79	51426,79	51426,79	51426,79	51426,79
	1	118075,8	118075,8	118075,8	118075,8	118075,8	118075,8
TSP	2	7066,498	7066,498	7066,498	7066,498	7066,498	7066,498
	3	44428,89	44428,89	44428,89	44428,89	44428,89	44428,89
	4	9415,178	9415,178	9418,576	9418,576	9418,576	9417,151
	0	6195,994	6195,994	6195,994	6195,994	6195,994	6195,994
	1	21695,58	21695,58	21695,58	21695,58	21695,79	21695,59
VRP	2	14439,79	14439,79	14439,79	14439,79	14439,79	14439,79
	3	8301,908	8301,908	8301,908	8301,908	8301,908	8301,908
	4	15388,98	15388,98	15388,98	15388,98	15388,98	15388,98

Tabel 3. Hasil Eksekusi VNS-HC

Permasalahan	Ins	Min	Q1	Median	Q3	Maks	Rata-Rata
	0	76	76	79	79	88	78,6129
	1	70	70	70	70	72	70,12903
SAT	2	68	68,5	73	81	82	74,48387
	3	26	26	26	26	26	26
	4	50	50,5	52	53	53	51,83871
	0	0,016965	0,016967	0,01697	0,020719	0,022096	0,018396
	1	0,016642	0,016648	0,016657	0,016658	0,021036	0,016799
Bin Packing	2	0,028275	0,028425	0,028425	0,030387	0,284251	0,040913
	3	0,026618	0,026618	0,026618	0,026629	0,027599	0,026809
	4	0,006139	0,006139	0,006982	0,007226	0,012342	0,007473
	0	6469	6469	6469	6469	6469	6469
	1	6372	6372	6420	6420	6420	6398,323
Flow Shop	2	6473	6473	6473	6473	6473	6473
	3	6386	6386	6386	6390	6390	6387,29
	4	6506	6506	6506	6552	6552	6525,29
	0	3442	3477	3497	3500	3502	3487,806
	1	3165	3290	3290	3290	3290	3272,581
Personnel Scheduling	2	490	535	610	872,5	2080	917,0968
	3	25	37	46	50	68	44,41935
	4	37	46	54	61	260	75,41935
	0	51653,44	51653,44	51653,44	51653,44	51653,44	51653,44
	1	117399,1	117399,1	117399,1	117399,1	117420,8	117403,3
TSP	2	7132,697	7132,697	7132,697	7132,697	7132,697	7132,697
	3	44444,5	44444,5	44444,5	44444,5	44444,5	44444,5
	4	9349,059	9349,059	9349,059	9349,059	9349,059	9349,059
	0	6205,338	6205,338	6205,338	6205,338	6205,338	6205,338
	1	22703,79	22703,79	22703,79	22703,79	22703,79	22703,79
VRP	2	14412,67	14412,67	14412,67	14412,67	14412,67	14412,67
	3	6307,648	6307,648	6307,648	6307,648	6307,648	6307,648
	4	14326,73	14326,73	14326,73	14326,73	14326,73	14326,73

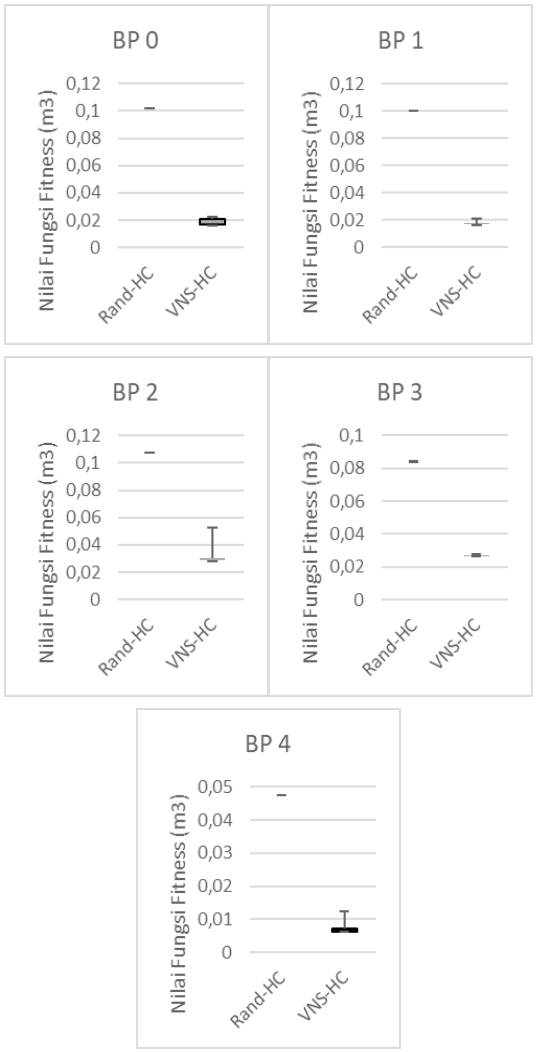
a. Perbandingan Penyebaran Data



Gambar 2. BoxPlot Kinerja Rand-HC dan VNS-HC pada Domain SAT

Perbandingan penyebaran data dari strategi *high level* berdasarkan nilai minimal, kuartil pertama, median, kuartil ketiga, dan nilai maksimal menggunakan diagram boxplot. Perbandingan penyebaran nilai fungsi fitness dari algoritma Rand-HC versus VNS-HC yang diuji coba pada keenam *problem domain* yang berbeda secara berurutan dapat dilihat pada Gambar 2 - Gambar 7. Perlu diperhatikan bahwa nilai fungsi *fitness* disini semakin kecil semakin bagus. Dari Gambar 2 - Gambar 7, dapat dilihat bahwa VNS-HC mengungguli Rand-HC hampir pada semua problem domain.

Selanjutnya, untuk membandingkan kedua algoritma lebih sistematis, untuk algoritma dengan nilai fungsi *fitness* lebih kecil akan diberikan satu poin. Jika nilai *fitness* yang dihasilkan sama, maka setiap algoritma diberikan satu poin. Poin yang dihasilkan akan dijumlahkan untuk mengetahui kinerja dari masing-masing algoritma. Karena ada 5 problem domain dan masing-masing domain ada 5 *instances*, maka jumlah maksimal skor untuk setiap problem domain optimasi adalah 25 poin, dan jumlah maksimal semua instance pada problem domain adalah 150 poin. Jumlah poin yang lebih besar menyatakan strategi tersebut memiliki kinerja yang lebih baik.



Gambar 3. BoxPlot Kinerja Rand-HC dan VNS-HC pada Domain BP

Secara ringkas, perbandingan Rand-HC dan VNS-HC terhadap masing-masing instance pada setiap problem domain optimasi, adalah sebagai berikut:

Pada permasalahan optimasi *satisfiability* (SAT), VNS-HC memiliki kinerja lebih baik daripada Rand-HC, terutama pada instance ke-1, ke-3, dan ke-4 seperti yang dapat dilihat pada diagram boxplot pada Gambar 2. Pada permasalahan SAT, VNS-HC unggul dengan total skor 22 poin sedangkan Rand-HC dengan total skor 3 poin.

Pada permasalahan optimasi bin *packing*, VNS-HC memiliki kinerja lebih baik pada semua *instance* dataset seperti yang terlihat pada *boxplot* seperti Gambar 3 dengan perbedaan yang sangat jauh Rand-HC. Pada permasalahan bin *packing*, VNS-HC unggul dengan total skor 25 poin sedangkan Rand-HC dengan total skor 0 poin.

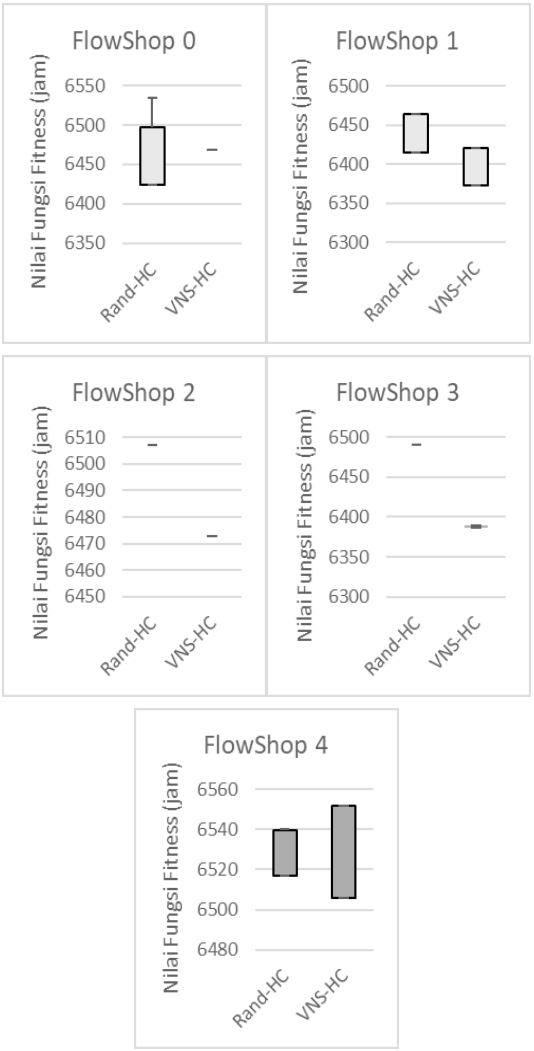
Pada permasalahan optimasi *flow shop*, VNS-HC memiliki kinerja lebih baik daripada Rand-HC pada

indeks *instance* dataset ke-1, ke-2, dan ke-3 seperti yang terlihat pada *boxplot* seperti Gambar 4. Pada permasalahan *flow shop*, VNS-HC unggul dengan total skor 21 poin sedangkan Rand-HC dengan total skor 4 poin.

Pada permasalahan optimasi personnel scheduling, VNS-HC memiliki kinerja lebih baik daripada Rand-HC pada beberapa nilai pada setiap *instance*, seperti nilai median pada *instance* ke-1 dan *instance* ke-2 seperti yang terlihat pada *boxplot* seperti Gambar 5. Pada permasalahan *personnel scheduling*, VNS-HC mendapatkan skor imbang dengan Rand-HC dengan total skor 13 poin.

Pada permasalahan optimasi TSP, VNS-HC memiliki kinerja lebih baik daripada Rand-HC pada *indeks instance dataset* ke-1 dan ke-4 seperti yang terlihat pada *boxplot* Gambar 6. Pada permasalahan ini, VNS-HC kalah dengan total skor 10 poin dibandingkan Rand-HC dengan total skor 15 poin.

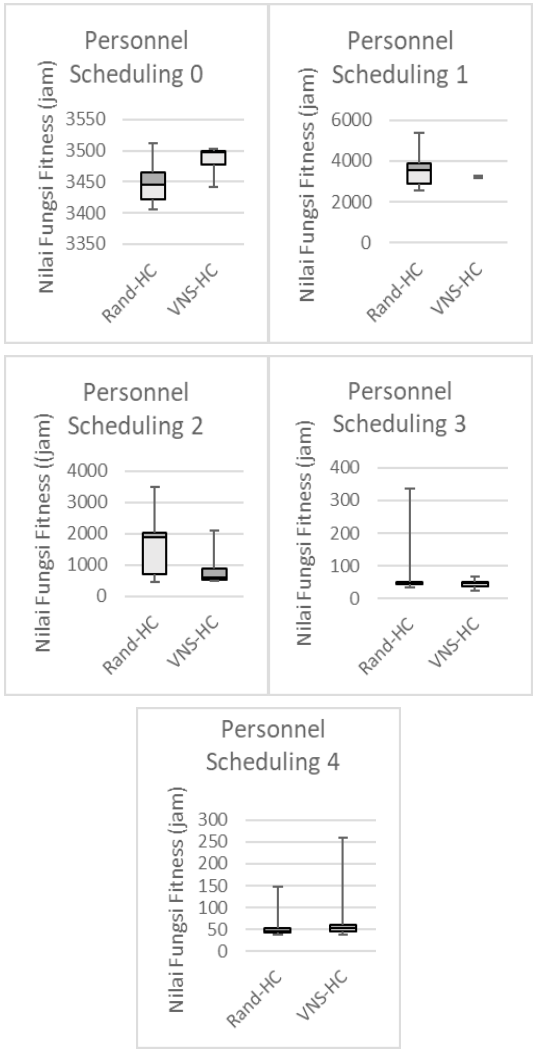
Pada permasalahan optimasi VRP, metode *hyper-heuristics* VNS-HC memiliki kinerja lebih baik daripada Rand-HC pada indeks *instance* dataset ke-2, ke-3, dan ke-4 seperti yang terlihat pada *boxplot* Gambar 7. Pada permasalahan ini, VNS-HC unggul dengan total skor 15 poin sedangkan Rand-HC dengan total skor 10 poin.



Gambar 4. BoxPlot Kinerja Rand-HC dan VNS-HC pada Domain FlowShop

Secara keseluruhan, pada diagram boxplot terhadap nilai minimal, kuartil pertama, median, kuartil ketiga, dan nilai maksimal hasil eksekusi, strategi VNS-HC memiliki kinerja yang lebih baik daripada Rand-HC dengan total skor 106, sedangkan Rand-HC dengan skor 45.

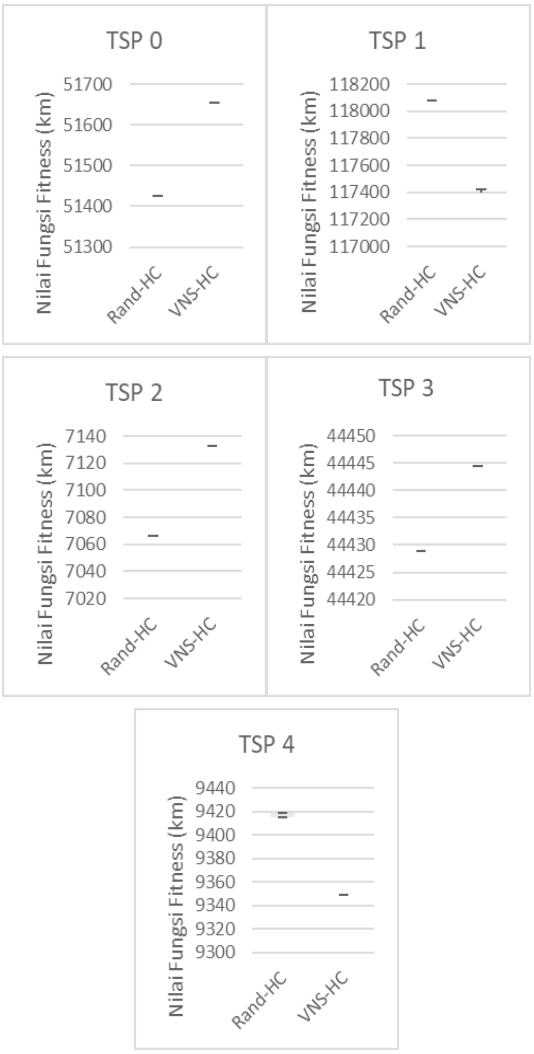
b. Perbandingan Nilai Median



Gambar 5. BoxPlot Kinerja Rand-HC dan VNS-HC pada Domain PS

Perbandingan nilai median untuk mengukur pemusatan data yang menjadi nilai tengah dari sekelompok data. Untuk melihat persaingan dari kedua metode tersebut, digunakan sistem poin FIFA. Hasil dari kinerja setiap strategi pada setiap permasalahan optimasi akan diberikan tiga (3) poin untuk strategi yang menang, satu (1) poin untuk hasil imbang, dan tidak ada (0) poin untuk strategi yang kalah. Menang, kalah, atau imbang tersebut dihitung dari penjumlahan peningkatan dan penurunan nilai pada kedua metode. Jika total dari peningkatan bernilai positif, maka metode VNS-HC dianggap menang dan diberikan tiga poin. Persaingan ini bertujuan untuk mencari strategi *high level heuristic* yang paling optimal untuk menyelesaikan masalah lintas domain. Berdasarkan

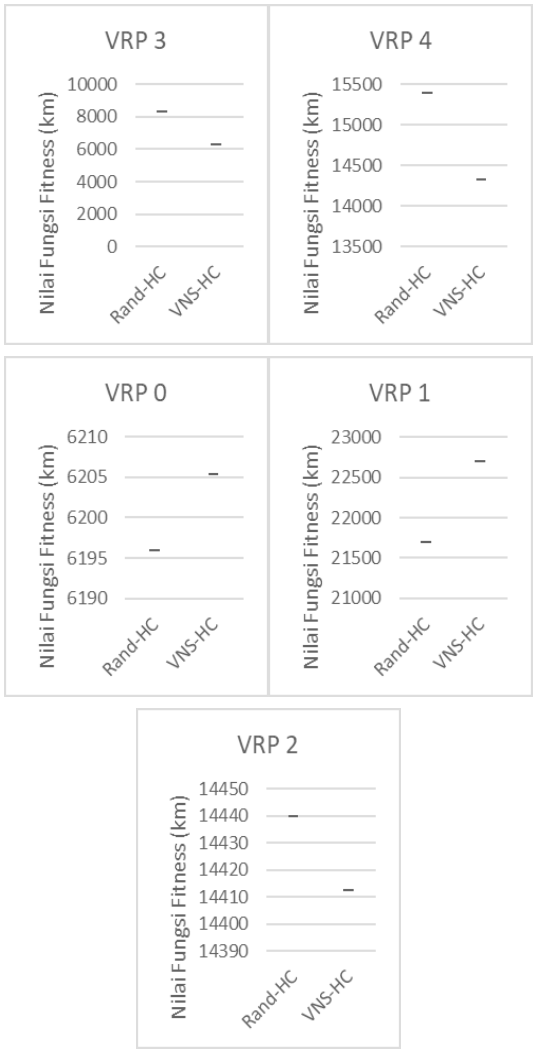
hasil perhitungan jumlah poin pada seluruh *instance* (30 *instance*), nilai median pada metode VNS-HC unggul pada lima permasalahan optimasi, yaitu pada permasalahan optimasi *SAT, bin packing, flow shop, personnel scheduling*, dan *VRP.* Sedangkan pada permasalahan optimasi TSP, metode VNS-HC mendapat poin seri dengan metode Rand-HC. Hal ini menyatakan bahwa strategi *high level* dengan metode seleksi *low level* secara sistematis, dapat meningkatkan kinerja *hyper-heuristics* dalam menyelesaikan masalah optimasi lintas domain. Untuk melihat grafik peningkatan dan penurunan kinerja pada setiap permasalahan optimasi, dapat dilihat pada Gambar 8.



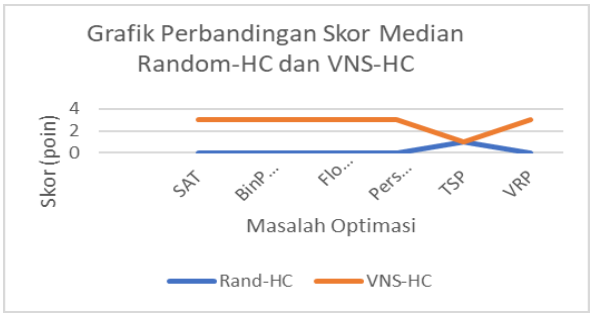
Gambar 6. BoxPlot Kinerja Rand-HC dan VNS-HC pada Domain TSP

c. Perbandingan Nilai Minimal

Perbandingan nilai minimal untuk menguji kinerja strategi high level heuristik dalam menghasilkan solusi optimal (fungsi *fitness* minimal). Untuk menguji kinerja setiap strategi dalam menemukan nilai fungsi *fitness* minimal, maka dilakukan perbandingan nilai minimal terhadap kedua metode dengan sistem poin yang sama dengan pengukuran nilai median.



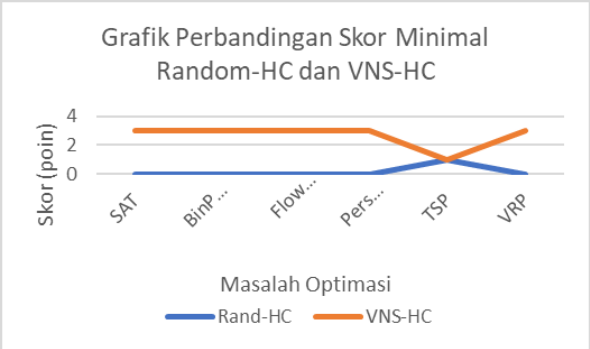
Gambar 7. BoxPlot Kinerja Rand-HC dan VNS-HC pada Domain VRP



Gambar 8. Grafik Perbandingan Skor Nilai Median

Berdasarkan hasil perhitungan jumlah poin pada seluruh *instance* (30 *instance*), nilai minimal pada metode VNS-HC secara keseluruhan unggul daripada strategi Rand-HC. VNS-HC unggul pada lima permasalahan optimasi, yaitu pada permasalahan optimasi SAT, *bin packing, flow shop, Personnel Scheduling*, dan VRP, sedangkan pada permasalahan optimasi TSP, strategi VNS-HC mendapat poin seri dengan metode Rand-HC. Hal ini menyatakan bahwa strategi high level dengen metode seleksi *low level heuristics* secara sistematis, dapat memberikan hasil yang lebih optimal dalam menyelesaikan

masalah optimasi lintas domain. Untuk melihat grafik peningkatan dan penurunan nilai minimal pada setiap permasalahan optimasi, dapat dilihat pada Gambar 9.



Gambar 9. Grafik Perbandingan Skor Nilai Minimal

5. Kesimpulan

Berdasarkan hasil eksperimen yang telah dilakukan dalam penelitian ini, pengujian terhadap seleksi LLH secara sistematis dengan mengadaptasi metode *Variable Neighborhood Search* yang dikombinasikan dengan metode move *acceptance* Hill Climbing sebagai strategi *high level heuristic* dapat meningkatkan kinerja dari *hyper-heuristics* yang diuji coba pada lintas enam problem domain optimasi dari HyFlex. Berdasarkan metode perbandingan yang berbeda dari enam permasalahan optimasi, terdapat empat permasalahan optimasi yang selalu mengalami peningkatan, yaitu SAT, *flow shop, bin packing*, dan VRP sedangkan pada domain lainnya mengalami penurunan dan tidak ada peningkatan. Secara ringkas dapat dikatakan bahwa algoritma baru yang diusulkan pada penelitian ini terbukti memiliki performa algoritma sebelumnya dari literatur.

Kenyataan bahwa algoritma VNS-HC hanya kalah pada problem domain TSP ini menarik untuk diinvestigasi pada penelitian berikutnya. Hal ini mungkin disebabkan struktur permasalahan TSP yang berbeda dari permasalahan yang lainya. Selain itu penggunaan algoritma *Hill Climbing* sebagai strategi *move acceptance* pada literatur terdahulu diketahui memang memiliki performa yang buruk, yaitu mudah terjebak pada lokal optima. Oleh karena itu pada penelitian selanjutnya algoritma VNS ini perlu digabungkan dengan algoritma yang lebih *robust*, sehingga dihasilkan performa algoritma yang lebih baik.

Daftar Pustaka

[1] R. Qu, "Meta-heurisitic Algorithm," *University of Nottingham*, Apr-2016. [Online]. Available: http://www.cs.nott.ac.uk/~pszrq.

[2] E. K. Burke, *Search methodologies*. New York: Springer, 2013.

[3] G. Ochoa, "Search-based Approaches and Hyper-heuristics."

[4] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of meta-heuristics*, Springer, 2010, pp. 449–468.

[5] E. K. Burke *et al.*, "Hyper-heuristics: a survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013.

[6] S. S. Choong, L.-P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Inf. Sci.*, vol. 436–437, pp. 89–107, Apr. 2018.

[7] M. Misir, W. Vancroonenburg, K. Verbeeck, and G. V. Berghe, "A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete," p. 11, 2011.

[8] W. G. Jackson, E. Ozcan, and J. H. Drake, "Late acceptance-based selection hyper-heuristics for cross-domain heuristic search," 2013, pp. 228–235.

[9] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *Eur. J. Oper. Res.*, vol. 130, no. 3, pp. 449–467, 2001.

[10] Z. Xu and Y. Cai, "Variable neighborhood search for consistent vehicle routing problem," *Expert Syst. Appl.*, vol. 113, pp. 66–76, Dec. 2018.

[11] Y. Qiu, L. Wang, X. Xu, X. Fang, and P. M. Pardalos, "A variable neighborhood search heuristic algorithm for production routing problems," *Appl. Soft Comput.*, vol. 66, pp. 311–318, May 2018.

[12] T. Czachórski, E. Gelenbe, K. Grochla, and R. Lent, Eds., "Performance of Selection Hyper-heuristics on the Extended HyFlex Domains," vol. 659, 2016.

[13] G. Ochoa *et al.*, "Hyflex: A benchmark framework for cross-domain heuristic search," in *European Conference on Evolutionary Computation in Combinatorial Optimization*, 2012, pp. 136–147.

[14] N. D. Angresti, A. Djunaidy, and A. Mukhlason, "Penerapan Hiperheuristik Berbasis Metode Simulated Penyelesaian Permasalahan Optimasi Lintas Domain," vol. 05, no. 01, p. 8, 2019.

[15] N. R. Sabar, M. Ayob, G. Kendall, and Rong Qu, "A Dynamic Multiarmed Bandit-Gene Expression Programming Hyper-Heuristic for Combinatorial Optimization Problems," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 217–228, Feb. 2015.

[16] A. Muklason, A. J. Parkes, E. Özcan, B. McCollum, and P. McMullan, "Fairness in examination timetabling: Student preferences and extended formulations," *Appl. Soft Comput.*, vol. 55, pp. 302–318, Jun. 2017.

[17] A. Muklason and A. J. Parkes, "Hyper-heuristics for Solving a Multi-objective Examination Timetabling Problem," *PATAT 2018 Proc. 12th Int. Conf. Pract. Theory Autom. Timetabling*, p. 1, 2018.