Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202183263
Vol. 8, No. 5, Oktober 2021, hlm. 871-878 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

PERBANDINGAN METODE PENYELESAIAN PERMASALAHAN OPTIMASI LINTAS DOMAIN DENGAN PENDEKATAN HYPER-HEURISTIC MENGGUNAKAN ALGORITMA *REINFORCEMENT LEARNING-LATE ACCEPTANCE*

Anang Firdaus[1], Ahmad Muklason*[2], Vicha Azthanty Supoyo[3]

[1,2,3] Institut Teknologi Sepuluh Nopember Surabaya
Email: [1]anangfirdaus05@gmail.com, [2]mukhlason@is.its.ac.id, [3]vichasupoyo@gmail.com
*Penulis Korespondensi

(Naskah masuk: 14 Februari 2020, diterima untuk diterbitkan: 15 Oktober 2021)

Abstrak

Sebuah organisasi terkadang membutuhkan solusi untuk permasalahan optimasi lintas domain. Permasalahan optimasi lintas domain merupakan permasalahan yang memiliki karakteristik berbeda, misalnya antar domain optimasi penjadwalan, rute kendaraan, bin packing, dan SAT. Optimasi tersebut digunakan untuk mendukung pengambilan keputusan sebuah organisasi. Dalam menyelesaikan permasalahan optimasi tersebut, dibutuhkan metode pencarian komputasi. Di literatur, hampir semua permasalahan optimasi dalam kelas NP-hard diselesaikan dengan pendekatan meta-heuristics. Akan tetapi meta-heuristic ini memiliki kekurangan, yaitu diperlukan *parameter tunning* untuk setiap problem domain yang berbeda. Sehingga pendekatan ini dirasa kurang efektif. Oleh karena itu diperlukan pendekatan baru, yaitu pendekatan hyper-heuristics. Metode hyper-heuristic merupakan metode pencarian komputasi approximate yang dapat menyelesaikan permasalahan optimasi lintas domain dengan waktu lebih cepat. Lintas domain permasalahan yang akan diselesaikan ada enam, yaitu satisfiability (SAT), one dimensional bin packing, permutation flow shop, personnel scheduling, travelling salesman problem (TSP), dan vehicle routing problem (VRP). Dalam meningkatkan kinerja, penelitian ini menguji pengaruh dari adaptasi algoritma Reinforcement Learning (RL) sebagai strategi seleksi LLH dikombinasikan dengan algoritma Late Acceptance sebagai move acceptance, selanjutnya disebut algoritma Reinforcement Learning-Late acceptance (RL-LA). Untuk mengetahui efektivitas performa dari algoritma RL-LA, performa algoritma RL-LA yang diusulkan dibandingkan dengan algoritma Simple Random-Late Acceptance (SR-LA). Hasil dari penelitian ini menunjukan bahwa algoritma yang diusulkan, i.e. RL-LA lebih unggul dari SR-LA pada 4 dari 6 domain permasalahan uji coba, yaitu SAT, personnel scheduling, TSP, dan VRP, sedangkan pada domain lainnya seperti bin packing dan flow shop mengalami penurunan. Secara lebih spesifik, RL-LA dapat meningkatkan peforma pencarian dalam menemukan solusi optimal pada 18 instance dari 30 instance atau sebesar 64%, dan jika dilihat dari nilai median dan minimum metode RL-LA lebih unggul 28% dari metode SR-LA. Kontribusi utama dari penelitian ini adalah studi performa algoritma hibrida reinforcement learning dan late acceptance dalam kerangka kerja hyper-heuristics untuk menyelesiakan permasalahan optimasi lintas domain.

Kata kunci: *Optimasi Lintas Domain, Hyper-heuristic, High level heuristic, Reinforcement Learning, Late Acceptance*

COMPARISON OF CROSS DOMAIN OPTIMIZATION COMPLETION METHOD FOR HYPER-HEURISTIC APPROACH USING REINFORCEMENT LEARNING-LATE ACCEPTANCE ALGORITHM

Abstract

An organization sometimes needs solutions to cross domain optimization problems. The problem of cross domain optimization is a problem that has different characteristics, for example between domain optimization scheduling, vehicle routes, bin packing, and SAT. This optimization is used to support an organization's decision making. In solving these optimization problems, a computational search method is needed. In the literature, almost all optimization problems in NP-hard class are solved by meta-heuristics approach. However, this meta-heuristic has drawbacks, namely tuning parameters are needed for each different problem domain. So this approach is considered less effective. Therefore a new approach is needed, namely the hyper-heuristics approach. Hyper-heuristic method is an approximate computational search method that can solve cross domain optimization problems faster. In this final project there are six cross domain problems to be solved, namely

satisfaction (SAT), one dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman problem (TSP), and vehicle routing problem (VRP). In improving performance, this study examines the effect of the adaptation of the Reinforcement Learning (RL) algorithm as LLH selection combined with the Late Acceptance algorithm as a move acceptance. The results of this study indicate that there are 4 out of six problem domains that have improved performance, namely the SAT, personnel scheduling, TSP, and VRP, while in other domains such as bin packing and flow shop has decreased.

Keywords: *Cross Domain Optimization, Hyper-heuristic, High level heuristic, Reinforcement Learning, Late Acceptance*

1. PENDAHULUAN

Permasalahan optimasi saat ini berkembang menjadi masalah yang kompleks karena besarnya jumlah input dataset dan batasan-batasan yang harus terpenuhi dalam menentukan optimalitas (Rego, C., Gamboa, D., Glover, F. & Osterman, C., 2011). Saat ini terdapat berbagai macam domain permasalahan optimasi kombinatorial, seperti permasalahan optimasi *vehicle routing problem* (VRP), *flow shop*, *bin packing*, dan penjadwalan. Sebuah organisasi terkadang juga membutuhkan banyak optimasi untuk mendukung bisnisnya. Sebagai contoh perusahaan yang bergerak dalam bidang logistik tidak hanya membutuhkan optimasi untuk satu domain, tetapi membutuhkan optimasi untuk banyak domain, yaitu optimasi penjadwalan sopir, rute kendaraan, dan pengemasan barang (*bin packing*) sehingga permasalahan ini dapat disebut dengan permasalahan lintas domain.

Dalam menyelesaikan banyak domain permasalahan optimasi dibutuhkan berbagai jenis metode atau algoritma pencarian komputasi. Metode atau algoritma pencarian dalam menyelesaikan masalah optimasi dikelompokkan menjadi dua kelompok, yaitu algoritma *exact* dan algoritma *approximate* (Ochoa, G., 2014). Algoritma *exact* digunakan dalam menyelesaikan permasalahan optimasi sederhana. Dalam menyelesaikan masalah sederhana, algoritma *exact* dapat menemukan solusi yang paling optimal dalam waktu singkat. Namun, pada permasalahan optimasi yang kompleks seperti TSP, rute kendaraan, dan penjadwalan dengan input dataset yang besar, algoritma *exact* belum mampu mencari solusi yang paling optimal dalam waktu singkat. Oleh karena itu, algoritma *approximate* seperti heuristik, *meta-heuristic*, *dan hyper-heuristic* digunakan sebagai pilihan dalam menyelesaikan permasalahan optimasi yang kompleks.

Algoritma *approximate* lebih mengutamakan pencarian yang cepat sehingga efisien daripada pencarian yang lengkap (*linear*). Hal tersebut menyebabkan hasil solusi yang ditemukan dengan metode ini bukanlah hasil yang paling optimal, tetapi cukup baik dan dapat diselesaikan dalam waktu singkat (*polynomial*).

Pengembangan metode heuristik adalah *meta-heuristic* yang menyelesaikan permasalahan optimasi secara spesifik berdasarkan karakteristik satu permasalahan tertentu (Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R., 2006). Dalam menyelesaikan banyak permasalahan optimasi, *meta-heuristic* harus mendeskripsikan dan memodelkan setiap domain permasalahan, serta menemukan metode pencarian yang tepat dalam menyelesaikan permasalahan setiap domain secara spesifik. Penyelesaian permasalahan optimasi secara spesifik membutuhkan biaya yang mahal dan waktu yang lama karena jika ada domain permasalahan baru, metode tersebut harus memodelkan masalah dan memodifikasi metode pencarian dari awal. Penerapan metode metode tersebut dapat memperlambat proses pengambilan keputusan untuk dalam menemukan solusi optimal untuk banyak permasalahan optimasi yang berbeda (lintas domain). Oleh karena itu, perlu suatu metode yang dapat menyelesaikan berbagai permasalahan optimasi secara *general* sehingga dapat mempercepat penyelesaian masalah dalam mencari solusi (Burke, E.K. & Kendall, G., 2013).

Hyper-heuristic adalah metodologi otomatis untuk mencari atau membuat heuristik dalam meningkatkan generalitas domain permasalahan dan instance berbagai domain permasalahan optimasi yang kompleks (Burke, E. K. , Hyde, M. , Kendall, G., Ochoa, G., Ozcan, E. & Woodward, J.R., 2006). Walaupun *hyper-heuristic* dapat meningkatkan generalitas pencarian komputasi pada banyak domain permasalahan optimasi, tetapi hasil pencarian *hyper-heuristic* tidak selalu optimal untuk semua domain permasalahan (Burke, E. K. & Kendall, G., 2013). Hal tersebut disebabkan oleh perbedaan karakteristik setiap domain permasalahan.

Strategi *high level heuristic* yang diusulkan, diuji coba pada *kerangka kerja Hyper-heuristics Flexible (HyFlex)*. Terdapat enam domain permasalahan kombinatorial pada framework HyFlex, yaitu *travelling salesman problem (TSP), vehicle routing problem (VRP) satisfiability (SAT), one dimensional bin packing (BP), permutation flow shop (FS)* dan *personel scheduling (PS),* (Ochoa, G., 2012).

Meta-heuristic merupakan algoritma yang dapat menyelesaikan masalah optimasi kompleks jika diselesaikan dengan algoritma eksak. Metode heuristik adalah metode yang digunakan untuk mencari solusi suatu masalah dimana solusi yang ditemukan merupakan *feasible solution* yang terbaik

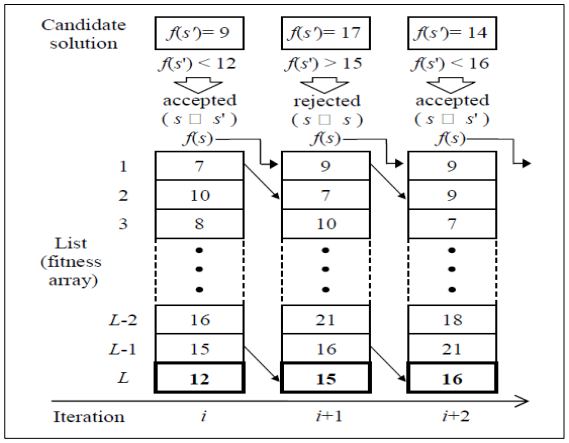
(Hillier, F. and Lieberman, G., 2010). Dalam pencarian solusi yang efisien dan komprehensif, metode *meta-heuristic* menggunakan mekanisme yang meniru perilaku sosial ataupun strategi yang ada di alam.

Algoritma *meta-heuristic* memiliki kecepatan pencarian solusi optimal yang lebih baik daripada metode tradisional (Madi, M., Markovi, D. & Radovanovi, M., 2013). Metode ini juga memberikan hasil yang lebih baik dibanding metode heuristik karena metode ini akan selalu berusaha untuk keluar dari solusi local optima. Meskipun tidak ada jaminan bahwa solusi yang ditemukan merupakan solusi yang optimal, metode *meta-heuristic* yang dibangun dengan baik dapat memberikan solusi yang mendekati solusi optimal (Szepesvári, C., 2010).

Reinforcement Learning (RL) merupakan sub area *machine learning* yang menitikberatkan pada bagaimana sebuah *agent* melakukan tindakan di lingkungannya. Dalam hal ini, *agent* berusaha untuk mengumpulkan *reward* sebanyak mungkin dalam waktu jangka panjang (Szepesvári, 2010). Dalam setiap iterasi langkah yang diambil, RL akan menentukan salah satu aksi dari banyak aksi yang mungkin dilakukan. Berikutnya, agent akan menerima reward atau punishment dari lingkungan atas aksi spesifik yang dilakukannya. Dalam hal ini, agent tidak mengetahu mana aksi terbaik yang harus dilakukan di beberapa *state*. Dengan demikian agent harus melakukan *trial and error* dengan melakukan beberapa aksi yang berbeda termasuk dengan urutan yang berbeda serta belajar dari pengalamannnya (Sutton and Barto, 2018).

Berbeda dengan tipe pembelajaran lainnya, RL lebih berfokus kepada *goal-directed learning* dan interaksi yang dilakukan dengan lingkungan. *Feedback* yang diberikan kepada *agent* bersifat evaluatif, sedangkan pada tipe pembelajaran lain, misalnya pada *supervised learning feedback* yang diberikan bersifat instruksional. *Agent* RL tidak diberitahu aksi mana yang benar maupun salah melainkan diberikan *reward signal* sesuai dengan aksi yang dilakukan, yang menandakan seberapa baik aksi yang telah diambil tersebut. Karenanya dibutuhkan eksplorasi dan pembelajaran yang bersifat *trial-and-error* agar Agent dapat bertindak sesuai dengan yang seharusnya.

Late Acceptance Strategy (LAS) adalah metode baru dalam metaheuristic yang memiliki strategi membandingkan antara kandidat solusi dengan salah satu solusi yang telah muncul pada beberapa iterasi sebelumnya (Özcan, 2009). Gambar 1 mengilustrasikan cara kerja algoritma LAS.



Gambar 1 Ilustrasi Iterasi Pada Eksekusi LAS

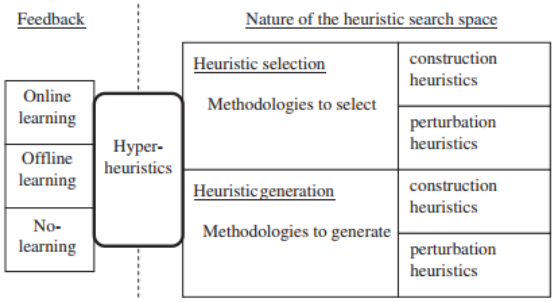
Hal tersebut menjadi pembeda antara LAS dengan metode pencarian metaheuristic lainnya seperti *Simulated Annealing* ataupun *Hill Climbing* yang memiliki strategi untuk membandingkan kandidat solusi dengan solusi terakhir secara langsung. Namun LAS termasuk dalam kelompok teknik pencarian perulangan meskipun memiliki tingkat mekanisme penerimaan yang lebih tinggi.

Dapat dilihat pada Gambar 1 bahwa kandidat solusi pada iterasi ke-i dan ke-(i+2) dapat diterima karena memiliki nilai yang lebih kecil dibandingkan dengan solusi pada urutan ke-L, sehingga kandidat solusi tersebut ditambahkan ke dalam daftar. Sedangkan untuk iterasi ke-(i+1), kandidat solusinya ditolak karena memiliki nilai yang lebih tinggi dibandingkan dengan solusi pada urutan ke-L, yang menyebabkan nilai solusi terakhir bernilai sama dengan nilai solusi sebelumnya.

Hyper-heuristic adalah sebuah metode yang penerapan pencariannya dilakukan pada *level* heuristik, yaitu dengan melakukan pemilihan dan menggunakan heuristik secara optimal agar dapat memecahkan suatu permasalahan (Burke, 2013). Pendekatan ini berbeda dengan pendekatan *meta-heuristic* biasa dimana ruang-pencariannya (search-space) berupa seluruh kemungkinan solusi, ruang pencarian hyperheuristic adalah semua kemungkinan low-level heuristic atau *low-level heuristic search space* yang dapat digunakan untuk memecahkan solusi (Harman, 2015). *Hyper-heuristic* melakukan proses pencarian pada domain permasalahan yang berbeda secara otomatis. Ide untuk mengotomatiskan desain heuristik telah ada pada tahun 1960, tetapi istilah *hyper-heuristic* baru diperkenalkan pada tahun 2000. *Hyper-heuristic* menggambarkan "heuristik untuk memilih heuristik" atau "heuristik untuk menghasilkan heuristik" (Burke & Kendall, 2013).

Hyper-heuristic memberikan pendekatan yang memiliki tujuan untuk melakukan desain metode heuristik untuk menyelesaikan permasalahan komputasional yang rumit. *Hyper-heuristic* mendeskripsikan penggunaan heuristik untuk memilih heuristik lainnya pada kasus optimasi

kombinatorial. Jadi, tujuan utama dari *hyper-heuristic* adalah untuk membuat desain metode umum, yang dapat memberikan solusi yang layak berdasarkan penggunaan dari LLH (*Low-Level Heuristic*).



Gambar 2 Klasifikasi Hyper-Heuristic

Sebagaimana diilustrasikan oleh Gambar 2, *Hyper-heuristic* terbagi menjadi dua dimensi klasifikasi, yaitu sifat ruang pencarian *hyper-heuristic* dan perbedaan *feedback* dari implementasi *low-level heuristics*. Berdasarkan dimensi sifat ruang pencarian, *hyper-heuristic* terbagi lagi menjadi dua, yaitu seleksi dan generasi *hyper-heuristic*.

Masing-masing seleksi dan generasi *hyper-heuristic* dibagi lagi berdasarkan sifat LLH sesuai dengan perbedaan antara paradigma pencarian konstruktif dan perturbatif. Konstruktif bekerja dengan mempertimbangkan solusi kandidat parsial (satu atau lebih komponen solusi hilang) dan secara iteratif membangun inisiasi solusi yang baru. Sementara itu, perturbatif mempertimbangkan inisiasi solusi lengkap dan sudah ada dan mengubahnya dengan memodifikasi satu komponen solusi atau lebih.

Hyper-heuristic terbagi menjadi 3 berdasarkan dimensi *feedback* sumber informasi pembelajaran, yaitu *online, offline*, dan *no-learning*. Dalam pembelajaran *hyper-heuristic online*, pembelajaran berlangsung ketika algoritma sedang menyelesaikan sebuah instance dari sebuah permasalahan, contohnya penerapan *reinforcement learning* pada seleksi heuristik, sedangkan dalam pembelajaran *hyper-heuristic offline*, mengumpulkan pengetahuan dalam bentuk peraturan atau program dari kumpulan *instance* pelatihan yang diharapkan akan menggeneralisasi untuk menyelesaikan kejadian yang tidak terlihat , contohnya penerapan *learning classifier system* dan *genetic programming* (Burke, 2006). Sedangkan *no-learning* tidak menggunakan pembelajaran apapun.

Hyper-heuristics flexible framework (HyFlex) adalah *software* kerangka kerja yang dirancang untuk memungkinkan pengembangan, pengujian, dan perbandingan algoritma pencarian heuristik tujuan umum iteratif seperti *hyper-heuristic*s. Kerangka kerja ini dibangun dengan menggunakan bahasa Java dan biasa digunakan oleh banyak peneliti (Ochoa, 2012).

HyFlex pertama kali digunakan untuk mendukung kompetisi penelitian internasional pada tahun 2011 dan dikenal sebagai CheSC (*Cross-Domain Heuristic Search Challenge*) 2011 (Ochoa, 2011). Di dalam Hyflex saat ini terdapat enam modul domain permasalahan yang diimplementasikan, yaitu *one dimensional bin packing*, *vehicle routing problem*, *permutation flow shop*, *personnel scheduling*, *traveling salesman problem*, dan *satisfiability*.

HyFlex telah ada pada Agustus 2010 saat peluncuran CheSC pada *International Conference on Practice and Theory of Automated Timetabling* (PATAT 2010). Saat ini telah banyak artikel implementasi *hyper-heuristic* yang dipublikasikan dengan HyFlex. *Cross-Domain Heuristic Search Challenge* bertujuan untuk melakukan pencarian dan pengoptimalan pada beberapa domain permasalahan (McCollum, 2011). *State-of-the-ar*t dari algoritma hyper-heuristics dapat dilihat di (Drake, 2020).

Penelitian sejenis sebelumnya dapat dilihat pada (Kumari, A.C. and Srinivas, K., 2016) yang menggunakan hyper-heuristics untuk menyelesaikan permasalahan klasterisasi modul software banyak tujuan dan (Hidayatul, 2019) untuk menyelesaikan permasalahan vehicle routing problem (VRP) banyak tujuan. Sedangkan penggunaan hyper-heuristics untuk menyelesaikan permasalahan penjadwalan dapat dilihat di (KUSUMAWARDANI, 2019), (MUKLASON, 2019a), dan (MUKLASON, 2019b). Untuk studi hyper-heuristik dalam menyelesaikan permasalahan optimasi lintas domain juga dapat dilihat di (DJUNAIDY, 2019) dimana dalam studi ini algoritma yang digunakan adalah *Variable Neighbourhood Search* (VNS).

2. METODE PENELITIAN

Metodologi dari penelitian ini dapat dijelaskan sebagai berikut:

2.1 Desain Algoritma

Tahap ini merupakan penggabungan antara *algoritma Reinforcement Learning* untuk melakukan seleksi LLH dan *Late Acceptence* (LA) sebagai penerima solusi yang terbaik sesuai dengan permasalahan yang telah didefinisikan pada bab pendahuluan. Pada tahapan ini akan dijabarkan metode yang akan digunakan untuk melakukan seleksi LLH dan mekanisme penerimaan solusi.

2.2 Implementasi

Pada langkah ini merupakan langkah implementasi desain algoritma yang sudah dibuat pada kerangka kerja HyFlex. Algoritma RL-LA yang telah didesain akan dibangun ke dalam HyFlex dalam bahasa pemrograman Java sehingga menjadi program yang siap digunakan untuk uji coba. Tahap ini dimulai dari persiapan *tools* hingga implementasi

program. Mekanisme algoritma *Reinforcement Learning* secara sederhana memberikan *reward* dan *penalty* pada setiap LLH.

Pada tahap awal, setiap LLH diberi skor awal yang sudah ditentukan. Jika LLH yang dipilih dan hasil solusinya dapat diterima, maka skor akan meningkat sampai skor mencapai batas yang ditentukan. Sebaliknya jika hasil ditolak, maka skor akan diturunkan hingga mencapai batas bawah yaitu nol. Pada setiap iterasi, LLH dengan skor tertinggi akan dipilih. Tetapi jika ada lebih dari satu LLH yang memiliki skor sama akan dipilih secara *random.*

Untuk mekanisme algoritma *Late Acceptence* sebagai move acceptance dasarnya sama seperti yang sudah dijelaskan pada tinjauan pustaka yaitu dengan membandingkan pada beberapa iterasi sebelumnya.

2.3 Uji Coba

Tahap ini merupakan tahap pengimplementasian dan eksperimen dari implementasi algoritma *hyper-heuristic* yang dilakukan. Uji coba dilakukan untuk mengetahui kinerja dari metode *hyper-heuristic* yang diimplementasikan.

Hasil dari uji coba digunakan untuk mencari strategi *high level heuristic* yang tepat untuk menyelesaikan masalah lintas domain. Tahap uji coba ini dilakukan pada kombinasi metode yang sesuai pada desain algoritma *hyper-heuristic*.

2.4 Analisa Hasil

Tahap analisis hasil dikerjakan setelah tahap implementasi dan uji coba pada HyFlex selesai. Analisis hasil digunakan untuk mengkur kinerja metode *hyper-heuristic*.

Nilai fungsi fitness menjadi nilai patokan yang akan dibandingkan setelah dieksekusi secara berulang kali (31 kali) dari setiap instance pada setiap domain permasalahan (total 30 *instance*) dengan waktu 60000 milidetik (10 menit).

Kriteria yang perlu diukur untuk membandingkan adalah nilai fungsi fitness terbaik (minimum), median, dan rata-rata, dan beberapa data statistik tambahan seperti nilai minimal, kuartil pertama, median, kuartil ketiga, dan nilai maksimum untuk setiap 30 instance domain permasalahan.

3. HASIL & PEMBAHASAN

Berikut ini dijabarkan hasil uji coba dan analisa yang telah dilakukan serta hasil desain dari algoritma Reinforcement Learning–Late Acceptance (RL-LA). Gambar 3 merupakan pseudocode dari desain algoritma RL-LA. Pengujian peforma dari metode yang diusulkan dalam mencari solusi optimal. Solusi optimal tidak dapat diukur nilai optimalitasnya karena metode ini adalah metode apporoximate.

```
Set ProblemDomain, DataInstance, Timelimit
Set parameter:L, upperBound, lowerBound, initialScore
Set panjang fitnestList sebesar L
Set initialScore pada setiap LLH
Generate solusiAwal
curSolution <-- solusiAwal
f0 <-- f(solusiAwal)
while (termination condition does not met)
    pilih LLH dengan score tertinggi, LLH*
    solusiBaru <-- apply LLH*(curSolution)
    f1 <-- f(solusiBaru)
    set x
    jika f1 < fitnestList(x)
        tambah score LLH*
        f0 <-- f1
        curSolution <--solusiBaru
    jika f1 > fitnestList(x)
        kurangi score LLH*
    tambahkan f1 ke fitnestList
Cetak f0
Cetak curSolution
```

Gambar 3. Desain Algoritma RL-LA

Namun, untuk memilih metode yang terbaik dalam mencari solusi optimal, perbandingan hasil uji coba dengan metode lainnya. Disini, perbandingan dilakukan dengan cara membandingkan metode RL-LA dengan metode SR-LA yang diimplementasikan ulang dalam menyelesaikan domain yang sama. Nilai median dan nilai minimum hasil eksekusi nilai fungsi objektif menjadi nilai perbandingan yang akan diuji. Perbandingan nilai median dan nilai minimum untuk melihat metode yang mengalami peningkatan. Metode dikatakan meningkat, jika nilai fungsi objektif lebih minimal. Untuk menghitung nilai peningkatan dengan cara seperti pada persamaan 1.

$$Nilai\ perubahan\ (\%) - ((\frac{a-b}{b}) \times 100) \qquad (1)$$

Nilai perubahan digunakan untuk melihat selisih peningkatan atau penurunan nilai fungsi objektif metode RL-LA setiap instance. Jika nilai perubahan bernilai positif, maka terjadi peningkatan. Namun, jika nilai perubahan bernilai negatif, maka terjadi penurunan peforma pencarian RL-LA. Instance yang mengalami peningkatan akan ditotalkan. Total instance yang meningkat dhitung persentase dengan cara seperti persamaan 2. Hasil pengujian nilai median dan nilai minimum metode SA dapat dilihat pada Tabel 2 dan Tabel 3. Nilai dengan cetak tebal pada tabel menunjukkan performa yang lebih baik pada setiap instance.

$$Instance\ terbaik\ (\%) = \left(\frac{instance\ min}{total\ instance} \times 100\right) \quad (2)$$

Tabel 1. Hasil Uji coba RL-LA

Domain	Instance	Min	Q1	Median	Q3	Max
SAT	3	5	8	10	13	28
	5	4	8	12	19	76

Domain	Instance	Min	Q1	Median	Q3	Max
	4	2	4	6	8	48
	10	8	11	14	21	56
	11	8	12	14	18	25
	7	0.16	0.17	0.18	0.19	0.19
	1	0.06	0.07	0.07	0.08	0.08
BP	9	0.04	0.04	0.04	0.05	0.05
	10	0.13	0.13	0.13	0.13	0.14
	11	0.08	0.08	0.09	0.09	0.09
	5	17	26	28	33.5	41
	9	9749	10087	10232	10765	22675
FS	8	3203	3324.5	3376	3438	3698
	10	1664	1955	2090	2219	3080
	11	320	385	410	447.5	1760
	1	6342	6394	6403	6413	6456
	8	26975	27021	27050	27057	2721
PS	3	6400	6429	6448	6462	6512
	10	11530	11580	11595	11613	11677
	11	26753	26817	26827	26868	26913
	0	49112	5045	50881	51352	53110
	8	21190321	22201046	23530137	25024672	25337433
TSP	2	7005	7081	7106	7196	7332
	7	6997	7078	7092	7123	7162
	6	55953	59775	60549	61391	64289
	6	90570	96548	99856	102477	109688
	2	15759	16784	17061	17948	18244
VRP	5	255161	316756	340975	348451	380042
	1	21916	24023	25123	26320	28589
	9	178289	197147	203517	215239	227753

Tabel 2. Pengujian Nilai Median RL-LA dan SR-LA

Domain	Ins	SRLA	RLLA	Delta	
	3	13	**10**	3	30%
	5	18	**12**	6	50%
SAT	4	9	**6**	3	50%
	10	15	**14**	1	7%
	11	14	14	0	0%
	7	0.18241	**0.179959**	0.00245	1.37%
	1	0.07678	**0.076116**	0.000670458	0.88%
BP	9	**0.04464**	0.049960	-0.005310	-10.63%
	10	0.12749	**0.127199**	0.000299	0.24%
	11	**0.07451**	0.085738	-0.01	-13.09%
	5	30	**28**	2	7.14%
	9	31713	**10232**	21481	209.94%
PS	8	3491	**3376**	115	3.41%
	10	**2015**	2090	-75	-3.59%
	11	**370**	410	-40	-9.76%
	1	6380	**6403**	-23	-0.36%
	8	**27000**	27050	-50	-0.18%
FS	3	**6441**	6448	-7	-0.11%
	10	**11562**	11595	-33	-0.28%
	11	**26798**	26827	-29	-0.11%
	0	51327.5	**50880.83**	446.6979	0.88%
	8	**22520346.6**	23530136.5	-1009789.94	-4.29%
TSP	2	**7080.01972**	7105.80593	-25.7862150	-0.36%
	7	75357.4912	**7092.303222**	68265.18801	962.52%
	6	**60192**	60549	-356	-0.59%
	6	108811	**99856**	8954	8.97%
	2	17978	**17061**	918	5.38%
VRP	5	360081	**340975**	19106	5.60%
	1	27486	**25123**	2363	9.40%
	9	224333	**203517**	20817	10.23%
Jumlah terbaik		11	18		
Persentase (%)		38%	62%		

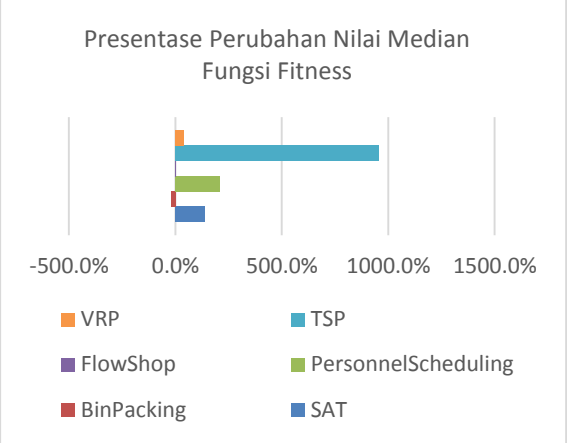
Tabel 3 Pengujian Nilai Minimum RL-LA dan SR-LA

Domain	Ins	SRLA	RLLA	Delta	
	3	7	**5**	2	40.00%
	5	9	**4**	5	125.0%
SAT	4	4	**2**	2	100.0%
	10	8	8	0	0.00%
	11	10	**8**	2	25.00%
	7	0.17368059	**0.15991**	0.013768	8.61%
	1	0.07109422	**0.06183**	0.009265	14.98%
BP	9	**0.03970**	0.04462	-0.00492	-11.04%
	10	0.12613	**0.12583**	0.000308	0.24%
	11	**0.06619**	0.07891	-0.01272	-16.12%
	5	19	**17**	2	11.76%
	9	10539	**9749**	790	8.10%
PS	8	3273	**3203**	70	2.19%
	10	**1558**	1664	-106	-6.37%
	11	320	320	0	0.00%
FS	1	**6340**	6342	-2	-0.03%

Domain	Ins	SRLA	RLLA	Delta	
	8	**26903**	26975	-72	-0.27%
	3	6408	**6400**	8	0.13%
	10	**11517**	11530	-13	-0.11%
	11	**26743**	26753	-10	-0.04%
	0	49632.41	**49112.41**	519.99	1.06%
	8	**21189643**	21190320.6	-677.558259	0.00%
TSP	2	**7003.4058**	7004.93190	-1.52609152	-0.02%
	7	69915.8509	**6996.70862**	62919.14227	899.2%
	6	**55139.3688**	55952.7020	-813.333179	-1.45%
	6	102113.8192	**90569.70358**	11544.11562	12.75%
	2	15878.64252	**15759.30365**	119.3388703	0.76%
VRP	5	332666.3338	**255160.9657**	77505.36814	30.38%
	1	22953.57458	**21916.35463**	1037.219947	4.73%
	9	200472.334	**178289.3463**	22182.98771	12.44%
Jumlah terbaik		10	18		
Persentase (%)		36%	64%		

Dari hasil perhitungan pada pengujian nilai median dan nilai minimum setiap *instance*, dapat dilihat penerapan metode RL-LA dapat meningkatkan peforma pencarian dalam menemukan solusi optimal pada 18 *instance* dari 30 *instance* dengan persentase 64%. Nilai median dan minimum metode RL-LA unggul 28% dari metode SR-LA. Secara keseluruhan, perbedaan peningkatan RL-LA cukup jauh. Namun, pada beberapa *instance* mengalami nilai imbang dan penurunan tetapi tidak signifikan. Metode RL-LA lebih baik daripada metode SR-LA untuk seleksi LLH. Akan tetapi, peningkatan dan penurunan tidak hanya diperngaruhi oleh mekanisme seleksi LLH saja, tetapi juga dipengaruhi oleh mekanisme penerimaan solusi. Kombinasi yang tepat dapat meningkatkan peforma dari hiperheuristik. Oleh karena itu, perlu melakukan banyak eksperimen untuk mengkombinasikan kedua mkanisme tersebut. Grafik pada Gambar 4 adalah perbandingan presentase perubahan total nilai median berdasarkan domain permasalahan. Nilai ini diperoleh dari membandingkan nilai fungsi fitness dari solusi yang dihasilkan oleh algoritma RL-LA dibandingkan dengan solusi yang dihasilkan oleh algoritma SR-LA.

Perubahan nilai fungsi fitness pada domain permasalahan bin packing dan flow shop mengalami perubahan negative, artinya RL-LA lebih buruk dibanding SR-LA, tetapi tidak terlalu signifikan. Sedangkan pada domain permasalahan SAT, Personnel Scheduling, TSP, dan VRP mengalami perubahan nilai positif yang tinggi, artinya RL-LA lebih baik dibanding SR-LA, khususnya pada permasalahan TSP, dimana RL-LA dapat menghasilkan solusi hampir 10 kali lebih baik (1000%) dari solusi yang dihasilkan oleh SR-LA.



Gambar 4. Perbandingan Presentase Perubahan Total Nilai Median Berdasarkan Domain Permasalahan

4. KESIMPULAN

Berdasarkan hasil studi performa algoritma RL-LA dengan algoritma pembanding SR-LA, dapat disimpulkan beberapa hal sebagai berikut:

1) Algoritma seleksi LLH dengan menggunakan *Reinforcement Learning* dan *Late Acceptance* sebagai *move acceptance* dapat meningkatkan kinerja hiperheuristik dalam menyelesaikan permasalahan optimasi lintas domain. Terdapat 4 dari enam domain permasalahan yang mengalami peningkatan kinerja, yaitu SAT, *personnel scheduling*, TSP, dan VRP, sedangkan pada domain lainnya seperti *bin packing* dan *flow shop* mengalami penurunan.
2) Seleksi LLH dengan menggunakan Reinforcement Learning pada dua domain yang telah diuji (bin packing dan flow shop) terjadi penurunan kinerja, tetapi penurunan yang terjadi tidak signifikan.

Untuk penelitian berikutnya, metode untuk mekanisme seleksi *low level heuristic* yang digunkan pada penelitian ini yaitu *Reinforcement Learning* dapat dikombinasikan dengan strategi *move acceptance* selain *late acceptance* seperti *great deluge* sehingga diharapkan dapat menghasilkan performa yang lebih bagus.

DAFTAR PUSTAKA

BURKE, E.K, HYDE, M., KENDALL, OCHOA, G., OZCAN, E., and WOODWARD, J. R., "Handbook of Metaheuristics," *Handb. Metaheuristics*, 2006.

BURKE, E.K. and KENDALL, G., *Search Methodologies*. 2013.

BURKE, E.K *et al.*, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013.

DJUNAIDY, A., ANGRESTI, N.D. and MUKHLASON, A., Hyper-heuristik untuk

Penyelesaian Masalah Optimasi Lintas Domain dengan Seleksi Heuristik berdasarkan Variable Neighborhood Search. Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika, 5(1), pp.51-60, 2019.

DRAKE, J.H., KHEIRI, A., OZCAN, E. and BURKE, E.K., 2019. Recent advances in selection hyper-heuristics. European Journal of Operational Research, 285 (2), p.405-428, 2020.

HARMAN, M. and CHICANO, F., "Search Based Software Engineering (SBSE)," *J. Syst. Softw.*, vol. 103, p. 266, 2015.

HIDAYATUL, Y. S., DJUNAIDY, A., & MUKLASON, A. Solving Multi-objective Vehicle Routing Problem Using Hyper-heuristic Method By Considering Balance of Route Distances. In 2019 International Conference on Information and Communications Technology (ICOIACT), pp. 937-942, 2019.

HILLIER, F. and LIEBERMAN, G., *Introduction to Operation Research*. 2010.

KUMARI, A.C. and SRIVINAS, K.. Hyper-heuristic approach for multi-objective software module clustering. Journal of Systems and Software, 117, pp.384-401, 2016.

KUSUMAWARDANI, D., MUKLASON, A. and SUPOYO, V.A., July. Examination Timetabling Automation and Optimization using Greedy-Simulated Annealing Hyper-heuristics Algorithm. In 2019 12th International Conference on Information & Communication Technology and System (ICTS),pp. 1-6. 2019

MADI, M., MARKOVI, D., and RADOVANOVI, M., "Comparison of Meta-Heuristic Algorithms for," *Facta Univ.*, vol. 11, no. 1, pp. 29–44, 2013.

MCCOLLUM, B. *et al.*, "The Cross-Domain Heuristic Search Challenge – An International Research Competition," 2011

MUKLASON, A., IRIANTI, R.G. and MAROM, A., Automated Course Timetabling Optimization Using Tabu-Variable Neighborhood Search Based Hyper-Heuristic Algorithm. Procedia Computer Science, 161, pp.656-664, 2019.

MUKLASON, A., SYAHRANI, G.B. and MAROM, A., Great Deluge Based Hyper-heuristics for Solving Real-world University Examination Timetabling Problem: New Data set and Approach. Procedia Computer Science, 161, pp.647-655, 2019

OZCAN, E., BYKOV, Y., BIRBEN, M., and BURKE, E.K., "Examination timetabling using late acceptance hyper-heuristics," *2009 IEEE Congr. Evol. Comput. CEC 2009*, pp. 997–1004, 2009.

OCHOA, G, M. H. G., "The Cross-domain Heuristic Search Challenge (CHeSC 2011)," 2011. [Online]. Available: http://www.asap.cs.nott.ac.uk/chesc2011/.

OCHOA, G. *et al.*, "HyFlex: A benchmark framework for cross-domain heuristic search," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7245 LNCS, pp. 136–147, 2012.

OCHOA, G. and N.D.., *Search-based Approaches and Hyper-heuristics*. 2014.

REGO, C., GAMBOA, D., GLOVER, F., and OSTERMAN, C., "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *Eur. J. Oper. Res.*, vol. 211, no. 3, pp. 427–441, 2011.

SUTTON, R. and BARTO, A., *Reinforcement Learning, An Introduction*. 2010.

SZEPESVARI, C., "Algorithms for Reinforcement Learning," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–103, 2010.