BAB 1
PENDAHULUAN

Bab ini akan menjelaskan latar belakang yang berisi penjabaran masalah yang akan diselesaikan pada penelitian, rumusan masalah, tujuan penelitian, batasan masalah penelitian, dan sistematika penulisan laporan untuk memberikan gambaran isi laporan secara keseluruhan kepada pembaca.

1.1 Latar Belakang

Optimasi adalah proses memilih solusi terbaik dari kumpulan kandidat solusi (Talbi, 2009). Optimasi identik dengan kata meminimalkan, memaksimalkan, efisiensi, optimal, ideal dan lain sebagainya. Jika sebuah permasalahan memiliki kata tersebut bisa dipastikan permasalahan tersebut adalah masalah optimasi. Optimasi sangat penting bahkan tanpa disadari kita melakukan optimasi. Contohnya ketika dalam perjalanan kita cenderung mencari jalan terdekat dan tidak macet untuk mencapai tujuan. Selain itu optimasi juga kita lakukan ketika melakukan penghematan biaya dan pengaturan *schedule* harian.

Optimasi juga sangat penting karena diterapkan diberbagai bidang keilmuan seperti:

- Bidang mekanika untuk optimasi gerak robot (*rigid body dynamic*) (Vukcevic, 2018).
- Bidang ekonomi untuk pengambilan keputusan ekonomi berdasarkan waktu (trading bit coin) (Dorfman, 1969).
- Bidang Elektro optimasi *active filter* (De, Kar, Mandal, & Ghoshal, 2015) dan komponent microwave (Koziel & Bandler, 2008).
- Bidang sipil pada resource levelling (Hegazy, 1999), daya tahan struktur bangunan (Piryonesi & Tavakolan, 2017) dan optimasi pengeluaran reservoir (Ahmadianfar, Adib, & Salarijazi, 2015).

- Bidang komputer pada pembuatan model *machine learning* dengan berusaha meminimumkan fungsi objektif (J.-R. Zhang, Zhang, Lok, & Lyu, 2007).

Karena optimasi sangat penting banyak peneliti berusaha menemukan algoritma untuk menyelesaikan permasalahan optimasi. Algoritma optimasi sangat banyak namun menurut (Talbi, 2009) algoritma optimasi bisa dibagi menjadi dua yaitu *exact* dan *approximate.* Algoritma optimasi *exact* merupakan algoritmaoptimasi yang selalu memberikan jawaban pasti namun memiliki waktu komputasi yang lambat. Hal ini disebabkan karena algoritma optimasi *exact* memeriksa semua kemungkinan solusi. Salah satu contoh algoritma optimasi *exact* adalah *brute force* (R. Agrawal & Srikant, 1994). Untuk meningkatkan waktu komputasi algoritma *brute force* dilakukan beberapa cara salah satunya dengan menggunakan *dynamic programming* (Stedinger, Sule, & Loucks, 1984) namun teknik terkadang sulit dilakukan pada permasalahan optimasi tertentu. Karena itu algoritma optimasi *exact* sangat cocok untuk permasalahan optimasi dengan lingkup permasalahan yang kecil. Algoritma optimasi *approximate* adalah algoritma optimasi yang memiliki waktu komputasi yang cepat namun tidak selalu memberikan jawaban pasti. Tidak selalu memberikan jawaban pasti karena hanya memberikan jawaban pendekatan atau terdapat unsur *random.*

Approximate bisa dibagi lagi menjadi dua yaitu *approximation* dan *heuristic. Approximation algorithm* adalah algoritma optimasi yang efisien yang dapat meperkirakan solusi terbaik dan pada inisialisasi tertentu dapat memberikan solusi global optima dan dapat dibuktikan (Williamson & Shmoys, 2011). Salah satu contoh algoritma *approximation* adalah algoritma *greedy*. Algoritma *greedy* dengan memilih local optima terbaik pada setiap iterasi dengan harapan mencapai global optima (Black, 2012). Kekurangan dari *approximation algorithm* adalah mudah terjebak pada lokal optima. Penyebabnya karena metode *approximation algorithm* terlalu *greedy* terlalu banyak eksploitasi namun kurang eksplorasi.

Metode *heuristic* bisa dibagi menjadi dua yaitu specific *heuristic* dan *metaheuristic. Specific heuristic* adalah algoritma optimasi yang terbatas pada permasalahan tertentu. Algoritma optimasi *metaheuristic* berasal dari dua kata yaitu meta yang berarti *high level* dan heuristic yang berarti pendekatan (Beheshti & Shamsuddin, 2013). Bisa disimpulkan *metaheuristic* adalah algoritma pendekatan tingkat tinggi. Disebut pendekatan karena nilai yang dihasilkan algoritma bukanlah nilai *exact* atau pasti dan disebut tingkat tinggi karena dapat mengatasi permasalahan algoritma *approximation* yaitu mudah terjebak lokal optima.

Secara garis besar algoritma *metaheuristic* bisa dibagi menjadi dua yaitu *singlebased* dan *populationbased* (Talbi, 2009). *Singlebased metaheuristic* melakukan evaluasi satu kandidat solusi lalu bergerak menggunakan *local search. Populationbased metaheuristic* melakukan evaluasi beberapa kandidat solusi lalu bergerak berdasarkan kandidat solusi lain.

Namun dari sekian banyak algoritma optimasi tidak semua peneliti yang memiliki permasalahan optimasi bisa memahami secara detail algoritma tersebut. Walaupun peneliti tersebut mengerti suatu algoritma belum tentu peneliti tersebut bisa melakukan *coding*. Oleh karena itu dibutuhkan *software library* untuk optimasi.

Sudah banyak sekali *library* optimasi yang menggunakan algoritma *populationbased metaheuristic*. Pada bahasa pemrograman python terdapat package NiaPy (Vrbančič, Brezočnik, Mlakar, Fister, & Fister Jr, 2018), C terdapat package LibOPT (Papa, Rosa, Rodrigues, & Yang, 2017) dan matlab melalui global optimization toolbox (Birge, 2003).

Pada bahasa pemrograman R juga terdapat banyak sekali *package* (*library* pada bahasa pemrograman R) optimasi menggunakan pendekatan *populationbased metaheuristic* seperti "DEoptim" (Ardia, Boudt, Carl, Mullen, & Peterson, 2011) menggunakan algoritma *differential evolution*, "hydroPSO" (Zambrano-Bigiarini, Rojas, & Zambrano-Bigiarini, 2018) menggunakan algoritma *particle swarm optimization*, "microbat" (Hwang & Moon, 2016) menggunakan *bat algorithm* dan "ABCoptim" (Vega & Muñoz, 2013) menggunakan *artificial bee colony algorithm*.

Meskipun *package* tersebut sudah tersedia, pada kasus tertentu pengguna R harus menggunakan beberapa *pack*age. Salah satu kasusnya, dalam optimasi sangat dianjurkan untuk mencoba berbagai macam algoritma optimasi untuk suatu permasalahan. Alasanya untuk mendapatkan hasil optimasi yang maksimal. Hal ini didasari oleh teorema "No Free Lunch" (Wolpert & Macready, 1997) yang mengatakan tidak ada algoritma optimasi yang bisa menyelesaikan semua permasalahan optimasi dengan sempurna. Namun menggunakan beberapa *package* tidaklah mudah karena *input* dan *output* yang berbeda.

Untuk menyelesaikan masalah tersebut terdapat R *package* "metaheuristicOpt" (Riza, Prasetyo, Iip, & Munir, 2018). MetaheuristicOpt menerapkan 11 algoritma *populationbased metaheuristic* yaitu *particle swarm optimization* (Eberhart & Kennedy, 1995), *ant lion optimizer* (Mirjalili, 2015b), *grey wolf optimizer* (Mirjalili, Mirjalili, & Lewis, 2014), *dragonfly algorithm* (Mirjalili, 2016a), *firefly algorithm* (Yang, 2009), Algoritma Genetika (Goldberg & Holland, 1988), *grasshoper optimization algorithm* (Saremi, Mirjalili, & Lewis, 2017), *moth flame optimizer* (Mirjalili, 2015a), *sine cosine algorithm* (Mirjalili, 2016b), *whale optimization algorithm* (Mirjalili & Lewis, 2016) dan *harmony search* (Geem, Kim, & Loganathan, 2001). *Input* dan *output* dari setiap algoritma pada R *package* "metaheuristicOpt" memiliki format yang sama (konsisten). Terlebih lagi terdapat fungsi utama, fungsi ini dapat memanggil beberapa algoritma dengan satu input sehingga pengguna tidak perlu memanggil beberapa algoritma dengan input yang sama.

Sebanyak 11 algoritma *metaheuristic* yang dimiliki R *package* "metaheuristicOpt" tidaklah banyak terhitung dari tahun 1975 sampai 2012 terdapat kurang lebih 31 algoritma metaheuristic (Beheshti & Shamsuddin, 2013). Selain itu juga terdapat teorema "No Free Lunch" yang menunjukan tidak ada algoritma optimasi yang bisa menyelesaikan semua permasalahan optimasi dengan sempurna. Ada kemungkinan besar algoritma lain diluar 11 algoritma yang dimiliki *package* "metaheuristicOpt" dapat melakukan optimasi lebih baik pada permasalahan tertentu. Selain itu dibandingkan dengan *software library* optimasi diluar R seperti NiaPy (27 algoritma) dan LibOpt (21

algoritma) R *package* "metaheuristicOpt" memiliki variansi algoritma yang sedikit. Selain itu menurut (Riza et al., 2018) beberapa algoritma pada R *package* "metaheuristicOpt" memiliki kompleksitas yang tinggi. Kompleksitas yang tinggi menyebabkan semakin tinggi iterasi semakin lambat algoritma tersebut. Padahal iterasi yang tinggi sangat penting untuk mendapatkan optimasi yang baik. Algoritma yang memiliki kompleksitas tinggi pada R *package* "metaheuristicOpt" diantaranya *ant lion optimizer*, *dragonfly algorithm, grey wolf optimizer, firefly algorithm, moth flame optimizer, sine cosine algorithm* dan *grasshooper optimization algorithm*. Lalu kebanyakan algoritma pada R *package* "metaheuristicOpt" memiliki *hyperparameter* yang sedikit. Ini berarti jika algoritma tersebut melakukan optimasi pada suatu permasalahan optimasi mendapatkan *error* yang tinggi (jauh dari global optima) maka algoritma tersebut tidak memiliki cara mengurangi *error* dari permasalahan tersebut. Namun jika suatu algoritma memiliki *hyperparameter* yang banyak ketika melakukan optimasi mendapatkan *error* yang tinggi maka *error* tersebut bisa dikurangi atau bahkan dihilangkan dengan *tuning hyperparameter*. Algoritma yang memiliki sedikit *hyperparameter* pada R *package* "metaheuristicOpt" diantaranya *ant lion optimizer*, *dragonfly algorithm*, *grey wolf optimizer*, *grasshopper optimization algorithm*, *moth flame optimizer, sine cosine algorithm* dan *whale optimizer algorithm*.

Tujuan penulis yaitu mengembangkan R *package* "metaheuristicOpt" dengan menambahkan 10 algoritma yaitu *clonal selection algorithm* (Castro & Zuben, 2002), *differential evolution* (Das & Suganthan, 2011), *shuffled frog leaping* (Eusuff, Lansey, & Pasha, 2006), *cat swarm optimization* (Chu, Tsai, & Pan, 2006), *artificial bee colony algorithm* (Karaboga & Akay, 2009), *krill herd algorithm* (Gandomi & Alavi, 2012), *cuckoo search* (Yang & Deb, 2009), *bat algorithm* (Yang, 2012), *gravitational based search* (Rashedi, Nezamabadi-pour, & Saryazdi, 2009) dan *black hole optimization* (Hatamlou, 2013). 10 algoritma ini dipilih karena R *package* "metaheuristicOpt" adalah R *package* optimasi berbasis *population based metaheuristic* dan 10 algoritma yang disebutkan sebelumnya adalah algoritma optimasi berbasis *population based methaeuristic.* Penambahan 10 algoritma ini juga untuk menutupi kelemahan

algoritma yang dimiliki oleh R *package* "metaheuristicOpt" sebelumnya. Algoritma yang memiliki kompleksitas yang kecil diantaranya *differential evolution, artificial bee colony algorithm, cuckoo search, bat algorithm* dan *black hole optimization.* Algoritma yang memiliki *hyperparameter* yang banyak diantaraya *clonal selection algorithm, differential evolution, shuffled frog leaping, cat swarm optimization, krill herd algorithm* dan *gravitational based search.* Dengan menambahkan 10 algoritma ini diharapkan bisa menambah variansi dari R *package* "metaheuristicOpt". Penulis memilih mengembangkan *package* ini ketimbang membuat *package* baru karena R *package* "metaheuristicOpt" dibuat dengan konsep konsistensi *input* dan *output.* Jika penulis membuat *package* optimasi yang baru dikhawatirkan akan berkontribusi pada permasalahan R *package* yang dijelaskan sebelumnya. Selain itu algoritma baru yang ditambahkan akan menambah variansi dengan tidak mempersulit pengguna (*input* dan *output* konsisten). Algoritma *population based metaheuristic* dipilih dalam penelitian ini karena memiliki kecepatan komputasi yang cepat, tidak mudah terjebak lokal optima dan masih terus dikembangkan (*trending*). Bukti algoritma *population based metaheuristic* masih terus dikembangkan adalah algoritma *population based metaheuristic* yang terbaru ditemukan pada tahun 2019 yaitu *butterfly optimization algorithm* (Arora & Singh, 2019).

1.2 Rumusan Masalah

Permasalahan dapat dirumuskan dari latar belakang penelitian yang dijelaskan diatas adalah:

1. Bagaimana mengembangkan R *package* "metaheuristicOpt" untuk menyelesaikan permasalahan optimasi yang mengimplementasikan algoritma *populationbased metaheuristic*?
2. Bagaimana performa algoritma yang baru ditambahkan kedalam R *package* "metaheuristicOpt" dalam menyelesaikan masalah optimasi?

1.3 Batasan Masalah

Berdasarkan analisis kebutuhan dan metode yang digunakan dalam penelitian ini, berikut adalah ruang lingkup penelitian yang dilakukan:

1. Jenis permasalah optimasi yang dapat diselesaikan oleh *software*/*package* terbatas hanya untuk model permasalahan optimasi *continue*.
2. Permasalahan optimasi yang dapat diselesaikan hanya memiliki satu fungsi tujuan (*single-objective optimization*).
3. Dalam menguji performa algoritma hanya menggunakan 2 parameter yaitu *fitness* dan waktu eksekusi.
4. Penelitian ini terbatas pada tahap optimasi pada tahapan optimasi.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah:

1. Mengembangkan R *package* "metaheuristicOpt" untuk menyelesaikan permasalahan optimasi yang mengimplementasikan algoritma *clonal selection algorithm. differential evolution, shuffled frog leaping, cat swarm optimization, artificial bee colony algorithm, krill-herd algorithm, cuckoo search, bat algorithm, gravitational based search* dan *blak hole-based optimization*.
2. Menganalisis performa *software* yang dikembangkan dalam menyelesaikan permasalahan optimasi yang diberikan.

1.5 Sistematika Penelitian

Sistematika penulisan skripsi ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan pemanfaatan optimasi didalam berbagai bidang, Permasalahan dalam optimisasi, pengenalan definisi dan macam algoritma *population based metaheuristic,* pengenalan *R package* yang akan dikembangkan yaitu "metaheuristicOpt" algoritma yang digunakan dan algoritma yang akan ditambahkan dan metodologi penelitian.

BAB II KAJIAN PUSTAKA

Bab ini menjelaskan konsep-konsep dasar untuk penelitian yang terdiri dari penjelasan mengenai Bahasa pemrograman R, definisi optimisasi dan permasalahanya, definisi *metaheuristic* dan penjelasan algoritma *population based metaheuristic* yang akan ditambahkan kedalam R *package* "metaheuristicOpt" yang terdiri dari penjelasan latar belakang dan *pseudocode* algoritma.

BAB III METODOLOGI PENELITIAN

Bab ini berisi langkah-langkah dalam penelitian, model penelitian, penjabaran serta alat dan bahan yang digunakan untuk penelitian. Pada model penelitian dimulai dari Tahapan dimulai dari studi literatur dan pengumpulan data. Lalu penerapan algoritma pada R *package* yang pada setiap algoritma terdapat analisis, desain, coding dan testing. Berikutnya test performa menggunakan *test function* dan penarikan kesimpulan.

BAB IV HASIL PENELITIAN DAN PEMBAHASAN

Pada bagian ini akan dibahas secara mendalam mengenai permasalahan-permasalahan yang sudah diungkapkan dalam rumusan masalah. Adapun yang akan dibahas yaitu penjelasan permasalahan optimasi yang akan menjadi alat ukur pada proses pengujian, hasil implementasi algoritma metaheuristik pada *package* R, pengujian *package*, analisis hasil pengujian dan perbandingan software yang memiliki fungsi yang sama dengan R *package* yang dikembangkan pada penelitian ini.

BAB V KESIMPULAN DAN SARAN

Bab ini akan memaparkan kesimpulan yang merupakan jawaban atas pertanyaan pada rumusan masalah, dan saran yang merupakan kumpulan saran dan rekomendasi dari penulis untuk penelitian dan pengembangan selanjutnya.