BAB 5
KESIMPULAN DAN SARAN

Pada bab ini akan dipaparkan kesimpulan yang diperoleh dari penelitian yang dilakukan dan saran untuk penelitian mengenai pengembangan R *package* "metaheuristicOpt".

5.1 Kesimpulan

Berdasarkan hasil penelitian, pengujian dan analisis terhadap *package* yang dibuat, maka dapat ditarik kesimpulan sebagai berikut:

1. Dalam pengembangan R *package* "metaheuristicOpt" dimulai dengan tahapan analisis. Tujuanya untuk menjaga arsitektur dan menjaga konsistensi *package* sebelumnya. Dari hasil analisis dilakukan desain fungsi apa saja yang akan ditambahkan kedalam *package*. Setelah itu dilakukan *coding* berdasarkan desain. Setiap algoritma yang ditambahkan dilakukan *coding* berdasarkan *pseudocode* algoritma. Lalu dilakukan *testing* menggunakan *test function* untuk memastikan setiap algoritma bisa melakukan optimasi. Terakhir dilakukan *deployment* package ke CRAN. *Package* bisa dilihat dan diunduh melalui laman https://cran.r-project.org/web/packages/metaheuristicOpt/index.html
2. Perbandingan nilai optimum fungsi menunjukan algoritma *bat algorithm* mendapatkan nilai optimum terbaik pada 7 *test function* (f_1 sampai f_4 dan f_9 sampai f_{11}). *Fitness* terbaik f_9 dan f_{11} juga dicapai oleh fungsi `CSO()`.Fungsi `DE()` berhasil mendapatkan *fitness* terbaik pada 3 *test function* (f_6, f_{12} dan f_{13}). Untuk *test function* lainya *fitnees* terbaik f_5 oleh `GBS()`, f_7 oleh `SFL()` dan f_8 oleh `ClONALG()`.Walaupun ada beberapa fungsi seperti `ABC()`, `KH()`, `CS()` dan `BHO()` yang tidak mendapatkan *fitness* terbaik namun tetap dianggap berhasil karena algoritma tersebut bisa mendapat *fitness* yang lebih baik pada *test function* lain. Dari segi waktu eksekusi fungsi `DE()`, `ABC()`, `CS()`, `BA()` dan `BHO()` berdasarkan hasil eksperimen hanya memerlukan waktu kurang dari 1 detik untuk menemukan nilai optimum. Sedangkan fungsi lain membutuhkan waktu yang lebih lama ini disebabkan karena

algoritma tersebut terlalu banyak melakukan komparasi atau duplikasi kandidat solusi. Namun waktu eksekusi yang singkat tidak selalu menghasilkan nilai optimum terbaik. Jika dibandingkan dengan algorima sebelumnya yang diterapkan pada R *package* "metaheuristicOpt" beberapa algoritma baru terbukti memiliki *fitness* yang lebih baik. Semakin tinggi parameter iterasi maka semakin baik *fitness* algorima. Dampak parameter populasi bervariasi pada setiap algoritma. Berdasarkan uji coba dengan kasus real mayoritas algoritma lama seperti `PSO()`, `ALO()`, `GOA()`, `MFO()` dan `WOA()` mendapatkan *fitness* yang kecil sedangkan mayoritas algoritma baru seperti `DE()`, `ABC()`, `CS()`, `BA()` dan `BHO()` memiliki waktu eksekusi yang cepat kurang dari satu detik. Namun beberapa algoritma mengalami program *error* seperti `KH()` dan `CSO()`.

5.2 Saran

Adapun saran penulis untuk kepentingan penelitian atau pengembangan lebih lanjut adalah sebagai berikut.

1. *Parallel computing.* Teknik vektorisasi terbatas oleh algoritma yang diggunakan. Ada kalanya suatu algoritma yang sudah diterapkan metode vektorisasi kecepatanya tidak mengalami perubahan yang besar atau bahkan algoritma tersebut tidak bisa dilakukan vektorisasi. Oleh karena itu salah satu solusinya adalah *parallel computing.*
2. Menulis ulang kode *package* kedalam bahasa pemrograman *low level* seperti C. Hal ini dilakukan untuk menambah kecepatan seperti yang sudah dijelaskan pada no 1.
3. Membuat fungsi khusus seperti plot convergence fungsi untuk melihat peforma fungsi pada setiap iterasi. Contoh lain seperti fungsi validasi fungsi objektif. Setiap fungsi pada R *package* "metaheuristicOpt" memiliki parameter fungsi objektif. Fungsi objektif dibuat oleh pengguna. Fungsi validasi ini memeriksa apakah fungsi objektif yang dibuat pengguna bisa digunakan pada setiap fungsi pada R *package* "metaheuristicOpt" atau tidak. Jika

tidak tampilkan *error message* dari kesalahan fungsi objektif tersebut.

4. Menambah algorima lain untuk menyelesaikan masalah optimasi. Algoritma *metaheuristic* baru masih terus bermunculan. Dengan menambah algoritma baru menambah variansi algoritma dari *package* “metaheuristicOpt”.