BAB III LANDASAN TEORI

3.1. Ant Colony Optimization (ACO)

Algoritma Ant Colony Opmization adalah sebuah algoritma metaheuristic yang digunakan untuk pencarian secara mendalam dengan kualitas solusi yang lebih baik dibandingkan dengan metode pencarian konvensional biasa. Ant Colony biasanya digunakan untuk mencari jalur terpendek [11]. Menurut lopez [8] ACO adalah sebuah pendekatan *NP hard* yang mengkombinasikan masalah optimasi. Lopez mengatakan bahwa pada dasarnya ACO dibangun dari algoritma *Ant Colony System*(ACS) dan *MAX-MIN Ant System(MMAS).* ACO merupakan algoritma pencarian meteheuristik yang terinspirasi dari kelompok semut dalam mencari makanan [11] [7] [8].

Lebih lanjut Sutariya dalam penelitiannya menjelaskan bahwa "Ant Colony Optimization (ACO) is a class of optimization algorithms, inspired by an organized collaborative behavior of ants. Ants are creatures of nature with limited intelligence, which are wandering around their nests to forage for food"[1]. Jadi Ant Colony Optimization adalah sebuah rumpun ilu yang tergabung dalam swam intelligence dan merupakan bagian dari artificial intelligence.

Ant colony optimization (ACO) merupakan sebuah model yang dikembangkan dengan melihat semut sebagai objek utama pembentuk algoritmanya. Lebih lanjut Sutariya menjelaskan bahwa "The ant colony optimization algorithmic approach models the concept of food foraging, net

building, division of labour, cooperative support, self assembly and cemetery organization of real ants for the meta-heuristic approaches. ACO has been formalized in to a meta-heuristic computational approach" [14]. Untuk mendapatkan makanan, semut menggunakan cara tertentu untuk sampai pada tujuannya seperti dijelaskan sebagai berikut :

"While finding root from nest (source) to food (destination), ants communicate with other ants by depositing traces of pheromone (chemical substance) as they walk along their path. This indirect form of communication is called as stigmergy. As more ants travel over a particular path, the concentration of pheromone increases along that path. Pheromone along a path gradually evaporates, decreasing their concentration on that path. Among the multiple path between nest and food ants select the single optimal path on the basis of maximum pheromone concentration along the path and some heuristic functions". [14]

ACO merupakan pengembangan dari Ant Colony. Secara informal, ACO bekerja sebagai berikut: pertama kali, sejumlah *m* semut ditempatkan pada sejumlah *n* titik berdasarkan beberapa aturan inisialisasi (misalnya, secara acak). Setiap semut membuat sebuah tour (yaitu, sebuah solusi jalur evakuasi yang mungkin) dengan menerapkan sebuah aturan transisi status secara berulang kali. Selagi membangun tournya, setiap semut juga memodifikasi jumlah *pheromone* pada edge-edge yang dikunjunginya dengan menerapkan aturan pembaruan *pheromone* local yang telah disebutkan tadi.

Setelah semua semut mengakhiri tour mereka, jumlah *pheromone* yang ada pada edge-edge dimodifikasi kembali (dengan menerapkan aturan pembaruan *pheromone* global). Dalam membuat tour, semut 'dipandu' oleh informasi *heuristic* (mereka lebih memilih edge-edge yang pendek) dan oleh informasi *pheromone*.

Sebuah edge dengan jumlah *pheromone* yang tinggi merupakan pilihan yang sangat diinginkan. Kedua aturan pembaruan *pheromone* itu dirancang agar semut cenderung untuk memberi lebih banyak *pheromone* pada edge-edge yang harus mereka lewati. [9]. Algoritma ACO memiliki simulasi yang baik dalam memecahkan masalah optimasi [15].

Chaimongkon Chokpanyasuwan menjelaskan karakteristik ant colony sebagai berikut

"The characteristics of an artificial ant colony include positive feedback, distributed computation and the use of a constructive greedy heuristic. Positive feedback accounts for rapid discovery of good solutions, distributed computation avoids premature convergence and the greedy heuristic helps to find acceptable solutions in the early stages of the search process" [16].

Wang juga menjelaskan algoritma ACO sebagai "Ant colony optimization (ACO) goes through the necessary nodes on the graph to achieve the optimal solution with the objective of minimizing total production costs (TPC)" [6]. Mohammed dalam penelitiannya menjelaskan bahwa ACO dapat memberi solusi yang baik, penjelasannya sebagai berikut

"It provides a good diversification for the search in the solution space. Indeed, on the one hand it imposes lower and upper bounds on the amounts of pheromones to be deposited on the edges of the problem's graph. It thus mitigates the transition probabilities for the most taken edges. On the other hand, it allows, under the frequency of one iteration over two, to support the pheromones concentration on the edges of the best iteration's solution although it is not the best overall solution" [17].

Peneliti yang lain yakni Liqiang Liu memberi penjelasan yang lebih komprehensif yakni

"We can give such a model through the above process of analysis and expansion: assuming the food source is very where throughout in the continuous space, the quality of food source is different. At the initial moment, ants of ant colony distribute uniformly in the continuous space and release pheromones according to food sources of their position. The higher the quality of the food source, the more the pheromone ants released. The pheromone is distributed throughout the continuous space in a certain dispersed model, and ants perceive spatial concentration of pheromone intensity, moving to the position of a higher concentration of pheromone in a certain way and achieve the exploration of unknown regions during the move. The movement of the single ant will cause the change of the whole position distribution of ant colony, so that all the ants keep aggregating to the higher quality of food source and search the highest quality of food source in the continuous space eventually. This model is called position distribution model of ant colony foraging" [7]

Menurut Lopez [8], algoritma ACO pada awalnya dikonstruksikan dengan kandidat konstruksi untuk mendapatkan kombinasi masalah optimasi dimulai dengan solusi yang masih kosong. Kemudian ditambahkan dengan komponen solusi untuk mendapatkan kandidat solusi yang lebih menyeluruh dan optimal. Algoritma ACO dapat di ilustrasikan dengan langkah-langkah yang di tampilkan pada gambar 2 berikut ini :

```
SetParameters
Initialize(PhInfo)
best-so-far ← ∅
while termination condition not met do
   S ← ∅
   for each ant do
      solution ← ∅
      while solution not completed do
         component ← ConstrPolicy(PhInfo, HeurInfo)
         solution ← solution ∪ component
      end while
      S ← S ∪ {solution}
   end for
   ApplyLocalSearch( S )    # optional
   best-so-far ← FindBest( S ∪ {best-so-far} )
   B ← Select( S ∪ {best-so-far} )    # B ⊆ S ∪ {best-so-far}
   Evaporation( PhInfo, ρ )
   Update(PhInfo, B)
end while
return best-so-far
```

Figure 2 Ilustrasi algoritma ACO
(Sumber : Manuel Lopez et al, 2004)

Dalam *sigle objective ACO* yang disebut Lopez sebagai QAP, komponen solusi didefinisikan sebagai tanda fasilitas menuju sebuah lokasi. Karena itu, pheromone untuk menunjukan sebuah lokasi dapat mengikuti persamaan berikut :

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}.[\eta_{il}]^{\beta}} \quad if\ j \in N_i^k \quad \text{..... (1)}$$

Dimana :

$\tau_{ij}(t)$ = informasi pheromone pada iterasi yang sedang berjalan

η_{ij} = QAP spesifik pada informasi heuristic

$\alpha dan\ \beta$ = dua parameter yang menentukan nilai pheromone relative dan informasi heuristic

η_i^k = ant tetangga *(neighboorhod)* dari ant k.

$p_{ij}^{k}(t)$ = komponen solusi pada sebuah lokasi.

Untuk melakukan konstruksi solusi langkah tersebut diulangi sampai penentuan neighborhood menjadi kosong sehingga solusi lengkap dapat dihitung. Setelah solusi selesai dikonstruksi, pheromone kemudian di update. Performa evaporasi dilihat dengan penurunan pertama pheromone trails dengan factor yang konstan kemudian ant di ijinkan untuk mendeposit pheromone pada solusi yang telah dikonstruksikan. Setiap pheromone di update dengan mengikuti persamaan berikut :

$$\Delta\tau_{ij}(t+1) = \rho.\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \quad \text{..... (2)}$$

Dimana parameter ρ (dengan $0 \leq \rho < 1$) adalah jejak *persistence* (dimana 1 – ρ model evaporasi) dan $\Delta\tau_{ij}^{k}(t)$ adalah jumlah pheromone ant k yang ditambahkan kedalam pheromone trail. Dalam AS (ant system) jumlah tersebut didefinisikan sebagai persamaan (3) berikut ini :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/f(\phi^{k}) & if\ \phi_i^k = j\ \ in\ iteration\ t \\ 0 & otherwise \end{cases} \quad \text{..... (3)}$$

Dimana :

ϕ^{k} = kosntruksi solusi dengan ant k

$f(\phi^{k})$ = denotes nilai objek dari QAP untuk sebuah solusi ϕ.

Dengan demikian ACO diaplikasikan dalam jalur mitigasi dengan cara sebagai berikut :

- *Pertama* adalah dengan mengkonstruksikan masalah ke dalam sebuah graph G = (V, E) dengan V himpunan vertek yang merepresentasikan himpunan titik – titik, dan E adalah himpunan dari edge yang merepresentasikan jarak antara dua titik.
- *Kedua*, kendala yang terdapat pada jalur evakuasi yaitu mengunjungi n titik dengan titik-titik yang ada hanya dikunjungi sekali dimana titik awal sama dengan titik akhir. Tujuan dari pemilihan jalur terpendek, jalur logistic terpendek dan penentuan titik shelter yaitu mencari tour terpendek terhadap *n* titik.

- *Ketiga,* pemberian nilai intensitas jejak semut (*Pheromone*) dan informasi heuristik. Pemberian nilai *Pheromone*($rs\ \tau$) dalam jalur evakuasi dilakukan saat semut mengunjungi titik s setelah mengunjungi titik r. Informasi heuristik ( ) $rs\ \eta$ merupakan informasi yang merepresentasikan kualitas suatu edge antara titik r dan titik s, informasi ini dihitung sebelum algoritma dijalankan. Dengan $rs\ rs\ d\ \eta = 1$, $rs\ d$ adalah jarak antara titik r dan titik s.
- *Keempat,* (*tour construction*). Sebuah tour dibangun dengan mengaplikasikan prosedur sederhana sebagai berikut : Inisialisasi, ditempatkan m semut di n titik menurut aturan tertentu, kemudian semut mengaplikasikan *state transition rule* secara iteratif. Semut membangun lintasan sebagai berikut. Pada titik r, semut memilih secara probabilistik titik s yang belum dikunjungi menurut intensitas *Pheromone* ($rs\ \tau$) pada egde antara titik r ke titik s, serta informasi heuristik lokal yang ada, yaitu panjang sisi (egde). Semut secara probabilistik lebih menyukai titik yang dekat dan terhubung dengan tingkat *Pheromone* yang tinggi. Untuk membangun jalur terpendek yang mungkin, setiap semut mempunyai suatu bentuk memori yang disebut *tabu list*. *Tabu list* digunakan untuk menentukan himpunan titik yang masih harus dikunjungi pada setiap langkah dan untuk menjamin terbentuknya jalur terpendek yang mungkin. Selain itu semut bisa melacak kembali lintasannya, ketika sebuah lintasan itu diselesaikan. Setelah semua semut membangun sebuah tour, *Pheromone* di-*update* dengan cara mengurangi tingkat *Pheromone* oleh suatu faktor konstanta dan kemudian semut meletakkan *Pheromone* pada edge yang dilewati. *Update*

dilakukan sedemikian rupa sehingga edge dari lintasan yang lebih pendek dan dilewati banyak semut menerima jumlah *Pheromone* yang lebih banyak. Karena itu pada iterasi algoritma yang berikutnya akan mempunyai probabilitas yang lebih tinggi untuk dipilih.

3.2. Multiple Objective – Ant Colony Optimization (MO-ACO)

Multiple objective optimization didefinikan oleh Daniel Angus, et al sebagai optimisasi secara simultan dari fungsi multiple objective [18]. Dalam taxonomi MOACO, algoritma ini merupakan golongan algoritma metaheuristic yang memiliki fungsi colony lebih dari satu ant colony [19] [20]. Dengan demikian pheromonenya didapatkan dari lebih dari satu pheromone yang merupakan bagian dari colony masing-masing.

Algoritma MOACO menambahkan angka pada komponen algoritma kedalam algoritma *metaheuristic* ACO dengan menghubungkannya kedalam *multi-objective problem* [19]. Dengan mengasumsikan bahwa algoritma MO-ACO menggunakan *multi poheromone information*, dimana setiap informasi pheromone memiliki perbedaan yang mendefinisikan objek dan berat yang dimilikinya sehingga menjadi satu nilai saja. Nilai tersebut sama dengan berat yang digunakan dalam menghitung agregat object dalam skala pada *multi objective problem* [8].

Jika diasumsikan arbitrary *multi objective colony (MOCO) problem* dengan Q objective seperti yang dijelaskan Manuel Lopez [8], maka ant *k* ditambahkan

dengan *j* pada solusi partial akan mengikuti probability untuk menghitng pheromone dengan persamaan berikut :

$$p_{Sj}^{k} = \frac{\left[\prod_{q=1}^{Q}\left(\tau_{Sqj}^{q}\right)^{\lambda q}\right]^{\beta}}{\left(\left[\prod_{q=1}^{Q}\left(\tau_{Sql}^{q}\right)^{\lambda q}\right]^{\alpha}.\left[\prod_{q=1}^{Q}\left(\eta_{Sql}^{q}\right)^{\lambda q}\right]^{\beta}\right)} \qquad if\ j \in N_{S}^{k} \text{ ..... (4)}$$

Dimana :

S = {S_1,S_2,….., S_Q}, merupakan state factor pada *current partial solution*

S_q = vector yang dinilainya ditambahkan dengan each objective q=1,…, Q yang diberikan komponen solusi

N_S^k = *feasible neighborhood* dari ant *k* pada *current state vector*

τ_{Sqj}^{q} = Informasi pheromone dari j yang diberikan S_q untuk q^{th} *objective* pada *current iteration*

η_{Sqj}^{q} = informasi heuristic pada j didapatkan dari S_q yang terhubung dengan obejective q^{th}

Λ_q = nilai berat dari obejctive q^{th} dan dapat dilihat sebagai komponen q^{th} vector berat λ.

Pada performa terbaik ACO, Strategy update pheromone dapa single colony dilakukan dengan cara memilih satu solusi yang terbaik atau sedikit dari solusi yang terbaik yang ditetapkan dari *current iteration* atau sejak algoritmanya dimulai (*best so far strategy*). Strategy ini dilakukan dengan memili pheromone yang dominan dan direkomendasi untuk digunakan [19].

Strategy dalam melakukan update pheromone multi colony, hampir sama dengan strategi yang dipakai pada single colony. Untuk menjalankan spesialisai colony, setiap deposit pheromone hanya digunakan untuk satu colony saja [8]. Pemilihan pheromone dipilih yang dominan menggunakan metode *straightforwardly adapted* pada pendekatan multi colony.penggunaan ini merujuk pada *ant pereto set* yang kandidatnya didapatkan dari distribusi ant antara colony dan diijinkan melakukan deposit pheromone [8].

Pada penelitian ini, untuk mendapatkan mitigasi pada penentuan lokasi shelter pengungsi, jalur distribusi terbaik dan jalur evakuasi terbaik, maka peneliti menggunakan framework MOACO buatan dari Manuel López-Ibáñez and Thomas Stützle [21]. Framework ini memiliki langkah-langkah sebagai berikut :

1. Configurasi Framework MOACO

Untuk melakukan konfigurasi pada algoritma MOACO seperti yang dijelaskan Manuel Lopez [21] adalah :

> "Pertama, pada single objective ACO, informasi pheromone terkait dengan fungsi objektif yang adalah komponen solusi berkualitas tinggi dimana menerima lebih banyak pheromone. Dalam konteks multi-objective, fungsi objective adalah multi-dimensi dan tidak skalar, dan hanya ada urutan parsial antara solusi. Selain itu, dalam beberapa masalah, bisa masuk akal untuk memiliki komponen solusi yang berbeda untuk masing-masing objective, oleh karenanya, asosiasi matriks pheromonenya berbeda. Misalnya, dalam aplikasi ACO untuk multi objective dengan masalah penjadwalan, satu matriks pheromone dapat mewakili pekerjaan x hubungan pekerjaan dan posisi lainnya × hubungan pekerjaan. Dalam hal ini, kedua matriks dapat diperbarui dengan jumlah pheromone yang sama karena maknanya berbeda. Dalam masalah lain seperti BTSP, baik matriks pheromone mewakili komponen solusi yang sama (tepi), dan karenanya, pheromone perlu diperbarui baik menggunakan solusi yang berbeda atau menggunakan jumlah pheromone yang berbeda. Atau, bisa masuk akal juga menggunakan matriks pheromone tunggal. Jika beberapa pheromone atau matriks heuristik yang digunakan, biasanya dikumpulkan selama pembangunan solusi dengan cara bobot. Namun, ada perbedaan yang kuat antara algoritma MOACO dengan dilakukannya cara agregasi ini dan berapa banyak bobot yang digunakan. Sebuah pertanyaan desain lebih lanjut adalah berapa semut yang dipilih untuk menyetorkan pheromone. Algoritma MOACO memilih beberapa (atau semua) solusi nondominated, atau memilih solusi terbaik terhadap obejective yang terkait dengan matriks pheromone yang diperbarui. Akhirnya, beberapa algoritma MOACO menggunakan beberapa koloni semut. Kami mengambil pandangan bijaksana terhadap komponen dari desain algoritma MOACO, yang memungkinkan kita untuk memilih abstrak dari pilihan desain tertentu."

Manuel Lopez et all [21] telah mengidentifikasi beberapa komponen algoritmik yang sesuai dengan alternatif desain yang dijelaskan di atas. Mereka membagi komponen

ini ke dalam tiga kelompok utama: (i) yang berkaitan dengan definisi pheromone dan pembangunan solusi; (ii) yang berkaitan dengan update dari pheromone; dan (iii) yang berhubungan dengan penggunaan beberapa koloni. Lopez memeriksa komponen ini secara lebih rinci dalam bagian berikut.

a) Komponen MOACO Algorithm untuk konstruksi solusi

Menurut kajian yang dilakukan Manuel Lopez et all [21], komponen MOACO Algorithm dapat di konstruksi dengan solusi sebagai berikut :

"Salah satu keputusan desain yang paling penting ketika menerapkan algoritma ACO untuk sebuah masalah adalah definisi informasi pheromone (dan heuristik). Dalam konteks multi-objective, masalah ini diperparah oleh kenyataan bahwa ada beberapa cara untuk mengevaluasi solusi tunggal, dan oleh fakta bahwa beberapa solusi optimal mungkin (Pareto) diterapkan".

Menurut kajian Lopez et all [21] pertanyaan desain ini dapat dijawab oleh komponen algoritmik berikut :

Tunggal / Multi Pheromone / Informasi Heuristic. Lopez menjelaskan bahwa mungkin memiliki satu (tunggal) atau beberapa (multiple) matriks baik untuk informasi pheromone τ atau, jika berlaku, informasi heuristic η. Dalam kasus matriks multiple heuristik, setiap matriks dikaitkan dengan objective yang berbeda, seperti η^1 dan η^2 sesuai dengan informasi heuristik dari masing-masing objective. Untuk beberapa masalah, hal itu tidak mungkin untuk menentukan informasi heuristik untuk setiap objective secara independen, jadi kita akan dipaksa untuk menggunakan matriks heuristik tunggal. Demikian pula, dalam kasus matriks multiple pheromone, τ^1 dan τ^2 akan dikaitkan dengan masing-masing objective; prosedur untuk memilih solusi yang akan digunakan untuk memperbarui setiap matriks harus menegakkan perbedaan ini entah bagaimana [21].

Lebih lanjut lopez et all menjelaskan bahwa selama mengkonstruksi solusi, setiap kali kita memiliki beberapa matriks, kita perlu agregat matriks-matriks tersebut menjadi

matriks tunggal. Metode agregasi didefinisikan oleh komponen Agregasi, yang dijelaskan di bawah ini. Bisa dibayangkan untuk menggunakan metode agregasi yang berbeda untuk pheromone dan informasi heuristik; Namun, kita tidak mengeksplorasi kemungkinan ini di sini [21].

Aggregation. Nilai-nilai dari multi pheromone (atau heuristik) matriks harus dikumpulkan (aggregate) ke nilai pheromone tunggal (atau heuristik). Lopez et all [21] mengidentifikasi tiga alternatif dalam literature yaitu :

- Weighted sum :

  $\tau_{ij} = (1-\lambda)\tau_{ij}^1 + \lambda\tau_{ij}^2 \quad and \quad \eta_{ij} = (1-\lambda)\eta_{ij}^1 + \lambda\eta_{ij}^2.$ (5)

- Weighted product :

  $\tau_{ij} = \left(\tau_{ij}^1\right)^{(1-\lambda)}.\left(\tau_{ij}^2\right)^{\lambda} \quad and \quad \eta_{ij} = \left(\eta_{ij}^1\right)^{(1-\lambda)}.\left(\eta_{ij}^2\right)^{\lambda}.$ (6)

- Random :

  Pada setiap langkah pembangunan, diberi nomor acak seragam U (0, 1), semut memilih yang pertama dari dua matriks jika U (0, 1) < 1 - λ; selain itu memilih matriks lainnya.

Dalam tiga metode agregasi yang dijelaskan di atas, ada weight λ yang menjadi bias agregasi terhadap satu objective atau yang lain. Set bobot Λ didefinisikan oleh komponen $N^{weights}$ dan NextWeight.

$N^{weights}$ dan NextWeight. Himpunan bobot didefinisikan dalam interval [0, 1] sebagai

$$\Lambda = \{\lambda_i = 1 - (i-1)/N^{weights} - 1),\ \ i = 1, \dots, N^{weights})$$

dimana $N^{weights} = |\Lambda|$ adalah parameter dari framework.

Komponen NextWeight menentukan berat tertentu digunakan oleh semut pada iterasi tertentu. Opsi diuji untuk NextWeight yang baik bahwa semua semut menggunakan bobot

yang sama pada iterasi tertentu (satu-berat-per-iterasi), atau bahwa semua bobot yang digunakan pada setiap iterasi (all-bobot-per-iterasi). Dalam kasus satu-berat-per-iterasi, berat digunakan dalam iterasi yang berurutan berikut memerintahkan urutan unsur-unsur Λ, dan urutan terbalik ketika berat badan terakhir dalam urutan tercapai. Dalam kasus all-bobot-per-iterasi, ketika jumlah semut N^a lebih besar dari $N^{weights}$, beberapa semut akan menggunakan berat yang sama. Sebuah *speed-up* jelas adalah untuk menghitung agregasi dari matriks pheromone hanya sekali per berat per iterasi. Ini menjelaskan perilaku untuk koloni tunggal [21].

b) MOACO Algorithm untuk Pheromone Update

Untuk menjelaskan cara update pheromone, Manuel Lopez et all menjelaskan sebagai berikut :

"Mengingat satu set A^{upd} solusi calon memperbarui informasi pheromone, komponen PheromoneUpdate yang memutuskan solusi memperbarui informasi pheromone dan bagaimana. Parameter N^{upd} menentukan berapa banyak solusi digunakan untuk memperbarui setiap matriks pheromone" [21].

Menurut Manuel Lopez et all [21], untuk melakukan update pheromone, komponen PheromoneUpdate harus mempertimbangkan beberapa alternatif sebagai berikut:

- *Solusi Nondominated.* Solusi yang digunakan untuk memperbarui informasi pheromone adalah nondominated solusi dalam A^{upd}. Ketika ada lebih nondominated solusi dari N^{upd}, harus diterapkan mekanisme pemotongan dari SPEA2 untuk memilih hanya solusi N^{upd}. Pada prinsipnya, adalah mungkin untuk menggabungkan pheromone nondominated ini Metode update dan matriks multiple pheromone

dengan menggunakan solusi yang sama untuk memperbarui kedua matriks selama update berbeda di setiap matriks pheromone.

- *Best-of-objective*. Mekanisme ini pertama memilih dari A^{upd} yang memiliki N^{upd} solusi terbaik terhadap masing-masing obejctive. Di kasus matriks multiple pheromone, setiap matriks pheromone diperbarui menggunakan solusi N^{upd} terkait dengan objective yang sesuai. Jika tidak, $2 \cdot N^{upd}$ digunakan untuk solusi memperbarui matriks pheromone tunggal.
- *Best-of-Objetice-per-Weight*. Untuk setiap berat λ dan masing-masing objective, ada daftar solusi terbaik N^{upd} untuk objective yang dihasilkan menggunakan λ. Dalam kasus tertentu jika $\lambda = 0$, maka kita hanya tetap menggunakan daftar untuk objective pertama, dan kami melakukan hal yang sama untuk $\lambda = 1$ dan objective kedua. Bila menggunakan matriks multiple pheromone, setiap matriks diperbarui hanya menggunakan solusi dari daftar yang terkait dengan objective yang sama. Dalam kasus matriks pheromone tunggal, kedua daftar yang digunakan untuk update. Oleh karena itu, dalam kasus tertentu dari dua bobot, metode ini setara untuk best-of-objective. Metode best-of-objective -per-weight yang digunakan oleh algoritma mACO-1 dan mACO- 2, dan kami memasukkannya untuk kelengkapan. Namun, tidak jelas bagaimana pendekatan ini harus diperluas ke beberapa koloni, karena solusi mungkin dipertukarkan antara koloni dengan bobot yang berbeda.

Metode di atas tidak menjelaskan apakah A^{upd} terdiri dari best-so-far atau solusi iterasi terbaik, karena ini ditentukan oleh algoritma ACO mendasari syarat tertentu. Dengan demikian saat menggunakan algoritma MOACO, pheromone dapat diupdate menggunakan salah satu metode diatas berdasarkan jenis algoritmanya.

c) MOACO Algorithm untuk Multiple Coloni

Penelitian yang dilakukan Manuel Lopez [21] menunjukan banyaknya algoritma MOACO yang mengusulkan penggunaan beberapa koloni semut, menggunakan berbagai definisi tentang apa itu koloni. Salah satu Pendekatan yang populer adalah untuk menentukan dua koloni, masing-masing memiliki informasi pheromone sendiri. Koloni ini mungkin menjadi solusi yang digunakan untuk memperbarui informasi pheromone. Beberapa semut, yang kadang-kadang dikatakan milik "Extra" koloni, mengumpulkan informasi pheromone dari dua koloni lainnya. Menurut Manuel Lopez, pendekatan ini sebenarnya setara untuk menggunakan salah satu koloni dengan beberapa matriks pheromone, yang dikumpulkan dengan cara pembobotan yang berbeda menggunakan allweights- per-iterasi pilihan, dan tampaknya tidak tepat mengatakan bahwa semut milik koloni yang berbeda hanya karena mereka menggunakan bobot yang berbeda. Sedangkan arsitektur multi-koloni dari Iredi yang dijelaskan Manuel Lopez [21] mendefinisikan koloni sebagai sebuah kelompok semut yang terkait dengan informasi pheromone tertentu, sehingga semut dari setiap koloni membangun solusi hanya menurut informasi pheromone koloni mereka. Pada beberapa kasus informasi pheromone, setiap koloni memiliki dua matriks pheromone, satu untuk setiap tujuan. Ketika set dari bobot yang digunakan, masing-masing koloni menetapkan sendiri bobot Λ_c. Arsitektur multi-koloni ini memungkinkan kita untuk menggeneralisasi semua komponen sebelumnya. Oleh karenanya, Manuel Lopez mengadopsinya untuk dijadikan kerangka kerja (*framework)* algoritma MOACO. Selain itu, koloni bekerja sama dengan menggunakan arsip umum solusi nondominated untuk mendeteksi yang didominasi. Lebih lanjut kerjasama diberlakukan dengan bertukar solusi untuk memperbarui informasi pheromone . Pengaturan ini dikendalikan oleh Komponen

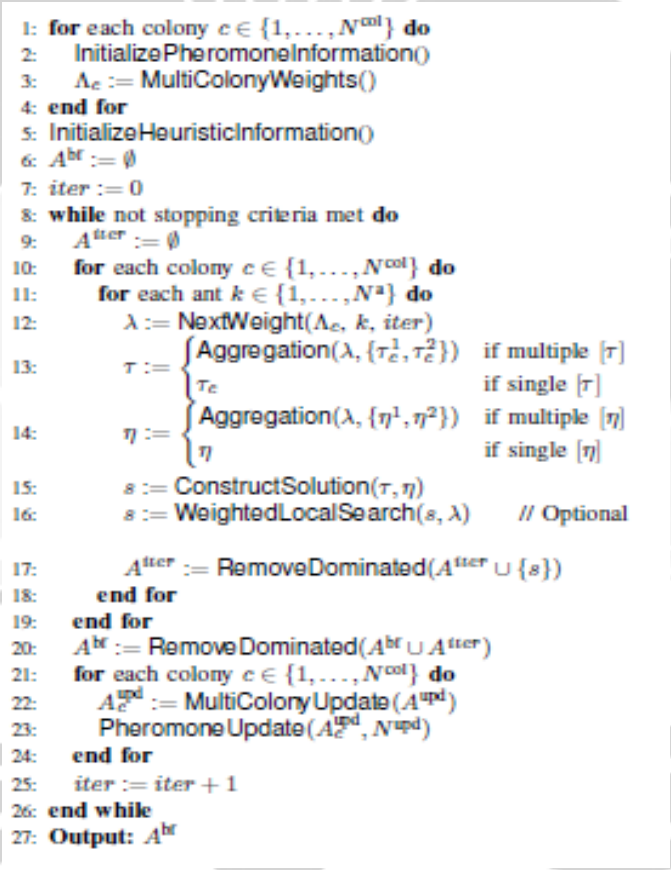
MultiColonyUpdate dijelaskan di bawah. Nomor koloni diberikan oleh komponen N^{col}, dan komponen dijelaskan di bawah hanya berpengaruh ketika N^{col} lebih besar dari satu.

MultiColonyWeights. Dalam kasus beberapa koloni seperti yang dijelaskan Manuel Lopez, diciptakan satu set bobot Λ_c dari ukuran $N^{weights}$ untuk setiap koloni c. Set ini dibangun dengan mengikuti dua alternatif : menguraikan dan interval yang tumpang tindih. Dalam kedua kasus, pertama dihasilkan jumlah yang diperlukan bobot yang merata dalam interval [0, 1]. Kemudian, untuk menguraikan interval, set ini dibagi menjadi sub interval yang menguraikan kondisi sama per koloni, yaitu c, $i = ((c - 1) \cdot N^{weights} + (i - 1)) / (N^{weights} \cdot N^{col})$, $i = 1,. . . , N^{weights}$, $c = 1,. . . , N^{col}$. Pada kasus interval yang tumpang tindih, sub-interval tumpang tindih dengan 50% dan karenanya, Λ_C dan Λ_{C+1} dibagi 50% dari bobot mereka.

MultiColonyUpdate. Manuel Lopez [21] menjelaskan bahwa dalam beberapa kasus koloni, solusi yang dihasilkan oleh semua koloni pada iterasi saat ini disimpan dalam arsip A^{iter}, sehingga semua koloni berkontribusi untuk mendeteksi dan menghapus solusi dominasi. Set A^{upd} kemudian dibangun dari solusi nondominated tersisa di A^{iter} atau dari arsip dari semua solusi nondominated pernah ditemukan, tergantung pada algoritma ACO yang mendasarinya. Sesudah ini langkah, solusi di A^{upd} ditugaskan kembali ke setiap koloni untuk memperbarui informasi pheromone . Metode dasar, disebut update dengan asal (*region*), memberikan setiap solusi dari A^{upd} ke koloni asli. Untuk menegakkan kerjasama yang lebih, memungkinkan solusi koloni pertukaran. Salah satu metode pertukaran, yang disebut update dengan wilayah, bekerja dengan cara membagi A^{upd} di bagian yang sama di antara koloni sedemikian rupa sehingga cara di setiap koloni sesuai dengan salah satu

wilayah ruang obyektif. Kedua pengaturan, update dengan asal dan pembaruan menurut wilayah.

d) MOACO Framework



1: **for** each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
2: InitializePheromoneInformation()
3: $\Lambda_c :=$ MultiColonyWeights()
4: **end for**
5: InitializeHeuristicInformation()
6: $A^{\text{br}} := \emptyset$
7: $iter := 0$
8: **while** not stopping criteria met **do**
9: $A^{iter} := \emptyset$
10: **for** each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
11: **for** each ant $k \in \{1, \ldots, N^{\text{a}}\}$ **do**
12: $\lambda :=$ NextWeight(Λ_c, k, $iter$)
13: $\tau := \begin{cases} \text{Aggregation}(\lambda, \{\tau_c^1, \tau_c^2\}) & \text{if multiple } [\tau] \\ \tau_c & \text{if single } [\tau] \end{cases}$
14: $\eta := \begin{cases} \text{Aggregation}(\lambda, \{\eta^1, \eta^2\}) & \text{if multiple } [\eta] \\ \eta & \text{if single } [\eta] \end{cases}$
15: $s :=$ ConstructSolution(τ, η)
16: $s :=$ WeightedLocalSearch(s, λ) // Optional
17: $A^{iter} :=$ RemoveDominated($A^{iter} \cup \{s\}$)
18: **end for**
19: **end for**
20: $A^{\text{br}} :=$ RemoveDominated($A^{\text{br}} \cup A^{iter}$)
21: **for** each colony $c \in \{1, \ldots, N^{\text{col}}\}$ **do**
22: $A_c^{\text{upd}} :=$ MultiColonyUpdate(A^{upd})
23: PheromoneUpdate($A_c^{\text{upd}}, N^{\text{upd}}$)
24: **end for**
25: $iter := iter + 1$
26: **end while**
27: **Output:** A^{br}

Figure 3 MOACO framework
(sumber : Manuel Lopez et all, 2012)

Tabel merangkum komponen algoritmik yang didefinisikan atas dan domain algoritma MOACO. Beberapa pengaturan hanya signifikan untuk nilai-nilai tertentu saja dari pengaturan lainnya. Sebagai contoh, sebuah agregasi Metode ini hanya diperlukan jika ada beberapa pheromone atau matriks heuristik. Kami mengusulkan Algoritma 2 sebagai

cara untuk mengintegrasikan komponen ini ke dalam kerangka kerja MOACO fleksibel untuk optimasi bi-objektif. Ini mengikuti garis dasar yang metaheuristik ACO. Pertama, algoritma menginisialisasi pheromone informasi (fungsi InitializePheromoneInformation, baris 2) dan himpunan bobot masing-masing koloni (fungsi MultiColonyWeights, jalur 3), sedangkan informasi heuristic diinisialisasi hanya sekali, karena hal itu dibagikan oleh berbagai koloni. Arsip semua solusi nondominated pernah ditemukan (terbaik-jadi-jauh arsip, Abf) dan counter iterasi yang diinisialisasi di jalur 6 dan 7, masing-masing. Pada setiap iterasi, setiap koloni membangun solusi Na menurut sendiri Informasi pheromone , yang mungkin mungkin digabungkan berat suatu? dari mengatur sendiri bobot (garis 13-15). Tergantung pada pengaturan dari NextWeight,? mungkin sama berat badan atau yang berbeda untuk setiap semut di koloni. dalam singleobjective algoritma ACO, solusi sering ditingkatkan dengan berarti dari pencarian lokal. Oleh karena itu, kami telah menyertakan opsional Prosedur WeightedLocalSearch (line 16) yang meningkatkan suatu solusi dengan menerapkan pencarian lokal untuk jumlah agregasi tertimbang fungsi objektif, menggunakan berat yang sama seperti itu digunakan untuk membangun solusi. Solusi baru ditambahkan ke arsip iterasi saat bersama oleh semua koloni Aiter (Line 17). Setelah semua semut dari seluruh koloni telah selesai membangun solusi, arsip Abf terbaik-jadi-jauh diperbarui dengan solusi nondominated ditemukan dalam iterasi saat Aiter (Fungsi RemoveDominated, line 20). Update dari pheromone terdiri dari dua tahap. Pertama, MultiColonyUpdate mendistribusikan arsip solusi untuk update (Aupd) antara koloni. Kedua, prosedur PheromoneUpdate (line 23) memutuskan bagaimana solusi ditugaskan untuk setiap koloni memperbarui Informasi pheromone koloni. Algoritma MOACO terus

sampai sejumlah iterasi atau batas waktu tercapai. Gambar 3 grafis menggambarkan hubungan antara berbagai komponen dari kerangka MOACO.

ALGORITHMIC COMPONENTS OF THE PROPOSED MOACO FRAMEWORK.

Component	Domain	Description
$[\tau]$	{ single, multiple }	Definition of pheromone matrices
$[\eta]$	{ single, multiple }	Definition of heuristic matrices
Aggregation	{ weighted sum, weighted product, random }	How weights are used to aggregate different matrices
$N^{weights}$	$\mathbb{N}^+$	Number of weights (per colony)
NextWeight	{ one weight per iteration (1wpi), all weights per iteration (awpi) }	How weights are used at each iteration
Pheromone Update	{ nondominated solutions (ND), best-of-objective (BO), best-of-objective-per-weight (BOW) }	Which solutions are selected for updating the pheromone matrices
N^{upd}	$\mathbb{N}^+$	Number of solutions that update each $[\tau]$ matrix
N^{col}	$\mathbb{N}^+$	Number of colonies
MultiColonyWeights	{ disjoint, overlapping }	How weights are partitioned among colonies
MultiColonyUpdate	{ origin, region }	How solutions are assigned to colonies for update

Figure 4 Komponen algoritma MOACO
(sumber : Manuel Lopaz at all, 2012)

2. Design MOACO Algorithm

Manuel Lopez [21] mendesain kerangka algoritma MOACO dengan cara konfigurasi kerangka tertentu dari meniru desain algoritma MOACO yang ada. Tujuannya adalah untuk menghasilkan algoritma yang lebih fleksibel dengan mengidentifikasi komponen konfigurasi algoritma tertentu dalam rangka penyesuaian pilihan desain algoritma dalam setiap kasus. Secara khusus, desain ini didasari oleh algoritma ACO, inisialisasi pheromone atau memperbarui jumlah, informasi heuristik, atau masalah-dependent speed-up.

Penelitian ini secara khusus mengadopsi desian yang dibuat oleh Manuel Lopez dalam mendesain algoritma MOACO. Hasil yang didapatkan, mengikuti pola framework MOACO yang dibuat oleh Manuel Lopez. Tabel II merangkum konfigurasi framework MOACO yang terdapat dalam ulasan ini.

TAXONOMY OF MOACO ALGORITHMS AS INSTANTIATIONS OF THE PROPOSED FRAMEWORK (N^A IS THE NUMBER OF ANTS).

Algorithm	N^{col}	$\|\tau\|$	$\|\eta\|$	Aggregation	$N^{weights}$	PheromoneUpdate	N^{upd}
MOAQ [4, 30]	1	1	2	–	$2\ (\Lambda = \{0, 1\})$	nondominated solutions	∞
BicriterionAnt [7]	any	2	2	weighted product	N^A	nondominated solutions	∞
MACS [6]	1	1	2	weighted product	N^A	nondominated solutions	∞
COMPETants [26]	1	2	2	weighted sum	$3\ (\Lambda = \{0, 0.5, 1\})$	best-of-objective	any
P-ACO [5]	1	2	1, 2	weighted sum	N^A	best-of-objective	2
mACO-1 [9]	1	2	2	random (τ) weighted sum (η)	$3\ (\Lambda = \{0, 0.5, 1\})$	best-of-objective-per-weight	1
mACO-2 [9]	1	2	2	weighted sum	$3\ (\Lambda = \{0, 0.5, 1\})$	best-of-objective-per-weight	1
mACO-3 [9]	1	1	1	–	–	nondominated solutions	∞
mACO-4 [9]	1	2	1	random (τ)	$1\ (\Lambda = \{0.5\})$	best-of-objective	1

Figure 5 Taksonomi algoritma MOACO
(sumber Manuel Lopez at all, 2012)

a) MOAQ

Manuel Lopez [21] menjelaskan bahwa meskipun beberapa Tujuan Ant-Q (MOAQ) awalnya dirancang untuk masalah dengan preferensi yang diberikan agar tujuan (optimasi leksikografis), tetapi secara umum MOAQ dirancang untuk masalah bi-objective dalam hal optimalisasi algoritma Pareto [22]. Garc'ıa-Mart'ınez menggunakan matriks pheromone tunggal dan ganda matriks heuristik, satu untuk setiap objective. Untuk pheromone update, mereka menggunakan semua solusi nondominated. Garc'ıa-Mart'ınez et al [22] membagi semut menjadi dua kelompok, dan setiap kelompok hanya menggunakan informasi heuristik sesuai dengan satu tujuan. Hal tersebut sama seperti Manuel Lopez dalam frameworknya yang dilakukan dengan menggunakan bobot $\Lambda = \{0, 1\}$ untuk menggabungkan *(aggregate)* dua matriks heuristik, sehingga setengah dari semut menggunakan setiap berat badan. Seperti satu set bobot efektif berarti bahwa tidak ada agregasi yang sebenarnya terjadi, tapi bukannya setengah dari semut menggunakan satu matriks heuristik dan setengah lainnya menggunakan yang lain [21].

b) bicriterionAnt

Salah satu bagian dalam algoritma MOACO yang dijelaskan Manuel Lopez [21] disebut BicriterionAnt, dengan multi pheromone dan matriks heuristik, digabungkan dengan produk tertimbang (*weighted*). Setiap k semut menggunakan bobot yang berbeda dengan λ_k untuk menggabungkan matriks pheromone; dengan demikian, ada banyak bobot sebagai semut ($N^{weights} = N^a$). Para penulis dari BicriterionAnt menyarankan untuk memperbarui matriks pheromone menggunakan solusi nondominated dalam iterasi saat ini. Selain itu, memperbarui kedua matriks pheromone dengan jumlah yang sama. Seperti update tidak menghasilkan matriks pheromone identik karena, dalam masalah tersebut ditangani oleh mereka, matriks pheromone merupakan komponen solusi yang berbeda seperti yang telah dijelaskan. Ketika menerapkan BicriterionAnt ke BTSP, Garc'ıa-Mart'ınez et al [22] menggunakan nilai fungsi tujuan dari masing-masing tujuan untuk memperbarui matriks yang sesuai $\Delta\tau^k = 1 / f^k(s_a)$. Manuel Lopez melakukan hal yang sama dalam rangka untuk kombinasi dari beberapa matriks pheromone dan pembaruan pheromone nondominated [21].

Selain algoritma di atas, diusulkan penggunaan beberapa koloni, dan menentukan koloni sebagai sekelompok semut yang membangun solusi menurut informasi pheromone mereka sendiri. Dalam hal ini, algoritma BicriterionAnt multi-koloni mirip dengan pendekatan multi-start dengan beberapa kekhasan. Pertama, koloni yang berbeda mengkhususkan diri di berbagai daerah perbatasan Pareto dengan menggunakan set yang berbeda dari bobot untuk agregat informasi pheromone . Kedua, koloni bekerja sama untuk mendeteksi solusi dominasi dengan menjaga solusi dalam arsip umum. Ketiga, koloni juga dapat bekerja sama dengan bertukar solusi. Sebagaimana dijelaskan dalam Bagian III-C, ini

cocok dengan pendekatan multi-koloni diadopsi dalam framework MOACO buatan Manuel Lopez [21].

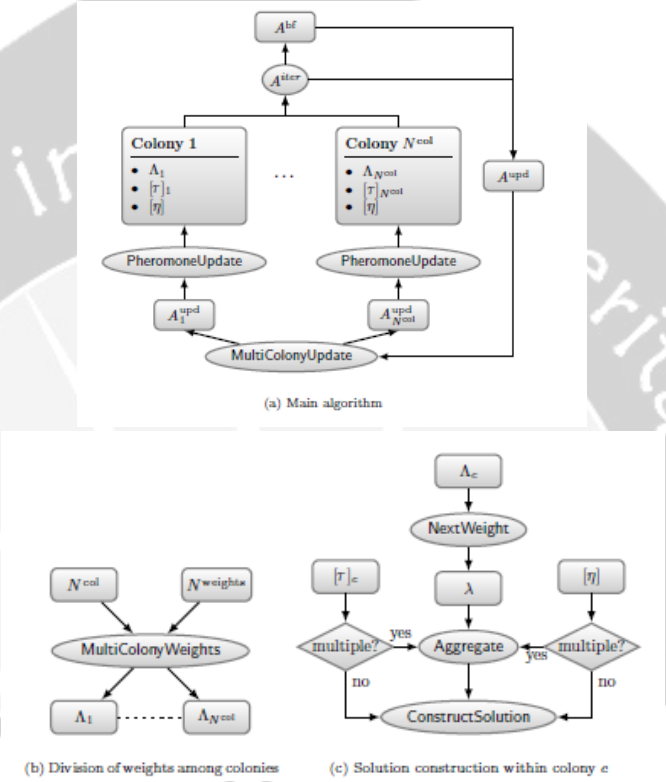


Figure 6 Gambar algoritma MOACO
(sumber: Manuel Lopez, 2012)

c) Multiple Ant Colony System

Beberapa Ant Colony System (MACS) seperti yang dijelaskan Manuel Lopez [21] menggunakan satu matriks heuristik untuk setiap tujuan dan matriks pheromone tunggal. Matriks heuristik dikumpulkan oleh produk tertimbang, dan setiap semut menggunakan berat yang berbeda. Selain itu, informasi pheromone diperbarui dengan solusi nondominated. MACS dapat dilihat sebagai varian MOAQ, seperti yang didefinisikan oleh Garc'ıa- Mart'ınez et al [22], yang menggunakan lebih dari dua bobot untuk mengumpulkan informasi heuristik. Di sisi lain, satu-satunya perbedaan antara MACS dan single-koloni

BicriterionAnt adalah jumlah matriks pheromone. Akibatnya, sangat mudah untuk mendefinisikan multi-koloni MACS, seperti yang Manuel Lopez lakukan dalam framework MOACO ciptaannya.

d) COMPETants

COMPETants seperti yang dijelaskan Manuel Lopez [21] disajikan sebagai pendekatan multi-koloni, dengan satu koloni untuk setiap tujuan. Setiap koloni memiliki satu pheromone dan matriks heuristik. Setiap koloni membangun solusi secara independen, kecuali untuk sejumlah semut (disebut "mata-mata"), yang agregat dua matriks pheromone oleh jumlah tertimbang (dengan $\lambda = 0,5$) baik pertama menggunakan atau matriks heuristik kedua, sehingga menciptakan dua solusi. Akhirnya, sejumlah semut dari setiap koloni digunakan untuk memperbarui matriks pheromone dari setiap koloni.

Manuel Lopez merumuskan COMPETants dalam rangka MOACO dengan menggunakan dua pheromone dan heuristik matriks, yang dikumpulkan oleh jumlah tertimbang dan tiga bobot $\Lambda = \{0, 0,5, 1\}$. Dengan demikian, COMPETants adalah pendekatan single-koloni dalam framework yang dibuat, dan itu sangat mudah untuk menentukan varian formulasi ini dengan jumlah sewenang-wenang koloni. Demi kesederhanaan, dalam framework MOACO, jumlah semut sama dibagi dengan jumlah bobot, dan jumlah semut per berat badan tidak berubah selama menjalankan, seperti dalam proposal awal; Namun demikian, fitur ini dapat ditambahkan ke kerangka sebagai komponen tambahan. Selain itu, setidaknya dalam TSP bi-obyektif, tidak pernah menggabungkan informasi heuristik, seperti dalam COMPETants asli, menyebabkan kualitas buruk di tengah depan Pareto. Oleh karena itu, kita tidak menangani khusus informasi heuristik untuk semut menggunakan $\lambda = 0,5$ tapi matriks heuristik dikumpulkan

dengan cara yang sama seperti matriks pheromone. Akhirnya, setiap matriks pheromone diperbarui dengan N^{upd} menghasilkan solusi terbaik untuk tujuan yang sesuai, yang merupakan *best-of-objective* pada pengaturan untuk PheromoneUpdate dalam framework MOACO.

e) Pareto Ant Colony Optimization

Pareto Ant Colony Optimization (P-ACO) seperti yang dijelaskan Manuel Lopez [21] menggunakan beberapa matriks pheromone, satu untuk setiap tujuan digabungkan dengan cara jumlah tertimbang. Seperti di BicriterionAnt dan MACS, berat yang berbeda terkait dengan setiap semut. Selain itu, matriks pheromone diperbarui dengan solusi yang terbaik dan kedua terbaik untuk masing-masing tujuan, yang pada dasarnya adalah sama dengan memperbarui metode yang digunakan oleh COMPETants, dan dalam framework MOACO sesuai dengan best-of-objective pengaturan untuk Pheromone-Update dan $N^{upd} = 2$. Dalam masalah yang dipecahkan, tidak ada definisi yang jelas tentang informasi heuristik untuk masing-masing tujuan, oleh karenanya, Manuel Lopez menggunakan matriks heuristik tunggal. Namun, RAPP menggunakan beberapa matriks heuristik, satu untuk setiap tujuan, yang digabungkan dengan cara yang sama sebagai matriks pheromone. Dalam pnelitian Lopez sebelumnya, menunjukkan bahwa ada perbedaan penting antara menggunakan satu atau dua matriks heuristik di P-ACO untuk TSP bi-objektif [23]. Secara khusus, Lopez mengamati bahwa matriks heuristik tunggal mengarah ke Pareto depan sangat sempit dan karenanya, framework MOACO menggunakan beberapa matriks heuristik ketika instantiating P-ACO. Meskipun demikian, framework MOACO bisa meniru kedua varian tersebut [21].

f) mACO Varian 1 (mACO-1)

Pada penelitian Manuel Lopez [21], terdapat usulan empat alternatif untuk desain algoritma MOACO. Varian pertama, mACO-1, digambarkan menggunakan beberapa koloni: satu koloni per obyektif, dan koloni tambahan yang membangun solusi dengan menggabungkan matriks pheromone dari dua koloni lainnya menyusul agregasi acak. Setiap koloni menggunakan informasi heuristik dari tujuan yang sesuai, sedangkan koloni ekstra agregat matriks heuristik. Dalam framework MOACO ini, Manuel Lopez merumuskan koloni tunggal dengan beberapa pheromone dan heuristik matriks dikumpulkan menggunakan tiga bobot $\Lambda = \{0, 0{,}5, 1\}$. Pada mACO-1 asli, informasi pheromone dari setiap koloni diperbarui dengan solusi terbaik yang dihasilkan oleh koloni tujuan yang sesuai dari koloni yang sama. Selain itu, algoritma solusi terbaik yang dihasilkan oleh koloni ekstra untuk setiap tujuan, dan menggunakan koloni untuk memperbarui matriks pheromone yang sesuai dari dua koloni lainnya. Karena setiap koloni dalam bahasa aslinya sesuai dengan berat yang berbeda λ dalam formulasi yang dihasilkan, metode pembaruan ini sesuai dalam framework untuk best-of-objective -per-berat dengan $N^{upd} = 1$.

g) mACO Varian 2 (mACO-2)

Varian kedua (mACO-2) memiliki sedikit perbedaan dengan mACO-1 yaitu hanya pada agregasi pheromone. Dalam mACO-2, menurut Manuel Lopez [21] matriks pheromone dikumpulkan dengan menjumlahkan matriks pheromone dari masing-masing tujuan. Ketika algoritma ACO yang mendasari adalah skala invarian, seperti AS, MMAS dan ACS, ini setara dengan weigth $\lambda = 0{,}5$. Manuel Lopez mengemukakan penelitian

sebelumnya pada TSP bi-tujuan menunjukkan bahwa tidak ada perbedaan yang signifikan dalam kualitas antara mACO-1 dan mACO-2 [23].

h) mACO Varian 3 (mACO-3)

Varian ketiga seperti yang dijelaskan oleh Manuel Lopez menggunakan matriks pheromone tunggal, yang diperbarui dengan menggunakan semua solusi nondominated (baik dari iterasi-terbaik arsip atau arsip terbaik-jadi-jauh) [21]. Pendekatan mACO-3 menekankan bahwa setiap nilai pheromone yang terkait dengan komponen solusi, diperbarui paling banyak sekali, meskipun banyak solusi menampungnya. Hal ini berbeda dengan algoritma lain seperti MOAQ, MACS, dan BicriterionAnt, yang menggunakan update pheromone nondominated. Namun, Manuel Lopez tidak menemukan keuntungan apapun dalam persyaratan khusus ini, karena kerangka ini tidak memasukkannya demi kesederhanaan. Informasi heuristik juga matriks tunggal. Dalam masalah di mana ada informasi heuristik tersedia untuk setiap tujuan, ini dikumpulkan sebelum menjalankan ke dalam matriks heuristik tunggal [21].

i) mACO Varian 4 (mACO-4)

Pada varian terakhir yang dijelaskan oleh Manuel Lopez [21], ada satu pheromone matriks per objektif, dan ini selalu digabungkan dengan cara acak sama seperti untuk m ACO-1, yang dalam prakteknya sesuai dengan agregasi acak dengan *single weight* $\lambda = 0{,}5$. Namun, ada matriks heuristik tunggal, seperti dalam mACO-3. Akhirnya, setiap matriks pheromone diperbarui dengan solusi terbaik untuk setiap tujuan, yang merupakan definisi best-of-objective pembaruan pheromone digunakan dalam kerangka kerja ini [21].

j) New Design Alternatif

Manuel Lopez [21] menjelaskan bahwa meskipun jumlah algoritma MOACO berbeda yang diusulkan dalam literatur, ada sejumlah yang jauh lebih besar dari kombinasi yang belum dijelajahi komponen algoritmik tersebut. Secara khusus, varian multi-koloni dapat didefinisikan untuk semua algoritma. Agregasi acak hanya diuji dengan $\lambda = 0,5$ sejauh ini. Selain itu, semua algoritma Ulasan atas penggunaan semua bobot yang tersedia di setiap iterasi (allweights-per-iterasi). Dalam penelitian yang dilakukan Manuel Lopez untuk bi-tujuan QAP, mengusulkan bahwa semua semut menggunakan bobot yang sama dalam satu iterasi, dan berat berikutnya dalam urutan di iterasi nex (satu-berat-per-iterasi) [21]. Oleh karena itu, dapat dengan mudah membangun varian baru yang paling algoritma pada Tabel II. varian baru seperti dapat dianggap algoritma MOACO baru.

Namun, harapannya bahwa banyak dari mereka tidak akan menyebabkan setiap terobosan signifikan. Di sisi lain, masalah tertentu yang sedang diselesaikan dapat mempengaruhi pilihan desain terbaik. Dalam kasus apapun, itu akan menjadi upaya besar untuk menguji mereka satu per satu untuk menemukan desain terbaik. Sebaliknya, Manuel Lopez mengusulkan untuk secara otomatis menemukan desain terbaik untuk masalah tertentu dengan (offline) konfigurasi otomatis dari *framework* MOACO [21]. Ada beberapa metode otomatis untuk konfigurasi offline algoritma optimasi single-objektif. Dalam penelitian Manuel Lopez [21] telah menambah Iterated F-Ras (I / F-Ras), untuk kasus multi-objective dengan menggunakan ukuran kualitas unary. Pada bagian berikutnya, kita menerapkan pendekatan ini untuk kerangka dijelaskan dalam makalah ini dan melakukan analisis rinci dari hasil.

3. Automatic Configuration MOACO Framework

Ada begitu banyak algormitma yang terkait dengan algortima MOACO ini, sehingga perlu dieksploitasi konfigurasi algoritma secara otomatis untuk mendapatkan varian MOACO yang efisien sesuai konteksnya. Dengan demikian diperlukan sebuah cara untuk menjadikannya otomatis. Tujuan otomatis (offline) konfigurasi ini adalah untuk menemukan algoiritma yang terbaik sesuai pengaturan parameter dari algoritma-algoritma yang ada untuk memecahkan masalah pada kasus-kasus yang ada. Framework yang dibuat Manuel Lopez bertujuan untuk menemukan Instansiasi yang baik dari metaheuristik sebuah ruang besar desain potensial. Selain itu, memperpanjang Ide untuk pertama kalinya dengan konteks multi-objective [21]. Framework tersebut menunjukkan bahwa dalam bagian ini terdapat konfigurasi otomatis dari framework MOACO yang fleksibel sehingga memungkinkan untuk menemukan hasil algoritma MOACO lebih baik untuk BTSP daripada yang tersedia dalam literatur.

Manuel Lopez juga mempelajari strategi konfigurasi yang berbeda, dan memeriksa hasil beberapa konfigurasi otomatis independen yang berjalan [21]. Metode konfigurasi otomatis yang digunakan seperti yang dijelaskan Manuel Lopez di sini adalah I / F-Ras, yang merupakan state-of-the-art otomatis metode konfigurasi yang mampu menangani terus menerus, kategoris dan parameter bersyarat. Manuel Lopez menggunakan pelaksanaan I / F-Ras yang disediakan oleh paket Irace. I / F-Ras bergantian antara menghasilkan konfigurasi calon baru dan melakukan balapan untuk membuang yang berkinerja terburuk. Dalam perlombaan, konfigurasi calon dijalankan pada satu Misalnya pada suatu waktu. I / F-Ras menggunakan tes Friedman diikuti dengan analisis post-test untuk membuang konfigurasi setiap kali ada bukti statistik yang cukup bahwa mereka melakukan lebih buruk

daripada yang lain. Ketika hanya sejumlah kecil dari konfigurasi tetap dalam lomba, lomba berhenti. Sebuah ras baru dimulai dengan konfigurasi terbaik sebelumnya ditemukan dan dengan calon baru konfigurasi yang dihasilkan dari konfigurasi terbaik menggunakan model probabilistik sederhana. Proses konfigurasi otomatis akan berhenti setelah mencapai angka (iterasi) maksimum yang diberikan (nomor dari berjalan atau batas waktu). Versi I / F-Ras saat ini dirancang untuk single objective masalah optimasi, dan karenanya, membutuhkan kriteria evaluasi yang memberikan nilai tunggal untuk masing-masing berjalan dari konfigurasi. Manuel Lopez menerapkan I / F-Race ke multi-objective dengan konteks cara pengukuran kualitas unary, yang menetapkan nilai kualitas tunggal untuk satu set nondominated. Manuel Lopez menguji dua langkah unary sebagai kriteria evaluasi I / F-Race, yaitu hyper volume1 dan (aditif) epsilon mengukur. Hyper volume adalah volume objective ruang lemah didominasi oleh satu set nondominated dan dibatasi oleh titik referensi yang ketat didominasi oleh semua Paretooptimal vektor obyektif. Semakin besar hypervolume tersebut, maka makin baik sesuai set nondominated. Aditif epsilon ukuran memberikan nilai minimum yang harus dikurangi dari semua tujuan dari satu set nondominated sehingga lemah mendominasi set referensi. Set referensi ini biasanya set nondominated dari semua solusi yang dikenal. Sebuah epsilon kecil Nilai ukuran lebih disukai [21].

Framework ini menggunakan algoritma ACO sebagai dasarnya, dan Manuel Lopez menggunakan MMAS sebagai ditetapkan untuk TSP [21]. Secara khusus, Manuel Lopez menggunakan pengaturan default yang diuraikan pada Tabel III, dengan menetapkan $\Delta\tau$ = 1 untuk Jumlah pheromone yang disimpan oleh semut, dan tidak menggunakan daftar calon untuk pembangunan solusi [21]. Manuel lopez juga menggabungkan pseudo-acak Aturan

pilihan aksi ACS, yang memungkinkan untuk greedier konstruksi solusi. Parameter q_0 mengontrol keserakahan tindakan aturan pilihan pseudo-random. Nilai dari $q_0 = 0$ menonaktifkan dan beralih kembali ke MMAS asli. Itu framework MOACO diimplementasikan dalam C, dan mendasari algoritma ACO berasal dari ACOTSP. Kode ini dikompilasi dengan gcc [21].

Manuel Lopez [21] membagi upaya konfigurasi dalam tiga tahap:

- Pertama, mempelajari apakah mungkin untuk secara otomatis menemukan desain MOACO baru. Oleh karena itu, hanya mengkonfigurasi komponen multi-objective dalam framework MOACO, dan membandingkan hasilnya dengan algoritma yang dijelaskan di dalam literatur.
- Kedua, menilai berapa banyak perbaikan mungkin dicapai dengan mengkonfigurasi pengaturan parameter algoritma ACO yang mendasari. Melakukannya dengan otomatis konfigurasi pengaturan algoritma ACO dari salah satu konfigurasi yang ditemukan pada tahap sebelumnya.
- Akhirnya, bertanya apakah konfigurasi dua-tahap ini memiliki pendekatan lebih baik daripada mengkonfigurasi semua komponen dan parameter sekaligus menggunakan upaya konfigurasi setara untuk dua tahap sebelumnya.

Parameter	Value
N^a	$24 \cdot \lfloor n/100 \rfloor$
ρ	0.02 ($n < 300$), 0.05 ($n \geq 300$)
q_0	0
α	1
β	2

Figure 7 Default parameter setting dari algoritma ACO (MMAS) (sumber: Manuel Lopez, 2012)

a) Konfigurasi Komponen Multiple Objective

Dari percobaan pertama yang dilakukan Manuel Lopez, tujuannya adalah untuk menemukan desain baru, mudah-mudahan lebih baik dari algoritma MOACO untuk BTSP. Manuel Lopez mencari desain dalam mode otomatis dengan mengkonfigurasi komponen multi-objective framework MOACO, sambil tetap tetap mendasari pengaturan parameter algoritma ACO. Pengaturan algoritma ACO tetap diberikan dalam Tabel III dan domain konfigurasi komponen multi-objective dijelaskan pada Tabel IV. Budget konfigurasi diatur 1000 set berjalan dari framework MOACO. Manuel lopez melakukan lima kali pengulangan independen dari proses konfigurasi menggunakan hypervolume sebagai kriteria evaluasi, dan lima pengulangan menggunakan ukuran epsilon unary [21].

Component	Domain	Constraint
$[\tau]$	{ single, multiple }	
$[\eta]$	{ single, multiple }	
Aggregation	{ weighted sum, weighted product, random }	only if multiple τ or η
N^{weights}	$\{2, 3, N^a/3, N^a/2, N^a\}$	(per colony)
NextWeight	{ one weight per iteration, all weights per iteration }	
PheromoneUpdate	{ ND, BO, BOW* }	* only with $N^{\text{col}} = 1$
N^{upd}	$\{1, 2, 5, 10\}$	
N^{col}	$\{1, 2, 3, 5, 10\}$	
MultiColonyWeights	{ disjoint, overlapping }	only if $N^{\text{col}} > 1$
MultiColonyUpdate	{ origin, region }	only if $N^{\text{col}} > 1$

Figure 8 Domain komponen algoritma MOACO
(sumber: Manuel Lopez, 2012)

Konfigurasi tertentu yang ditemukan diberikan dalam Tabel V dan VI untuk hypervolume dan ukuran epsilon masing-masing. Semua sepuluh konfigurasi menemukan penggunaan beberapa matriks heuristik, sejumlah besar koloni, update dengan wilayah dan

agregasi oleh produk tertimbang. Kebanyakan konfigurasi menggunakan beberapa matriks pheromone diperbarui oleh best-of-objective (BO), dan satu-weight-per-iterasi (1wpi).

Component	Run 1	Run 2	Run 3	Run 4	Run 5
$\|\tau\|$	single	multiple	multiple	multiple	multiple
$\|\eta\|$	multiple	multiple	multiple	multiple	multiple
Aggregation	w. product	w. product	w. product	w. product	w. product
$N^{weights}$	$N^3/3$	3	$N^3/2$	$N^3/3$	2
NextWeight	1wpi	1wpi	1wpi	1wpi	1wpi
PheromoneUpdate	ND	BO	BO	BO	BO
N^{upd}	1	2	2	2	1
N^{ants}	10	5	10	10	10
MultiColonyWeights	disjoint	overlap	disjoint	overlap	overlap
MultiColonyUpdate	region	region	region	region	region

Figure 9 Configurasi dengan lima tuning runs algoritma MOACO (sumber: Manuel Lopez, 2012)

Component	Run 1	Run 2	Run 3	Run 4	Run 5
$\|\tau\|$	multiple	single	multiple	multiple	multiple
$\|\eta\|$	multiple	multiple	multiple	multiple	multiple
Aggregation	w. product	w. product	w. product	w. product	w. product
$N^{weights}$	2	$N^3/3$	$N^3/2$	N^3	N^3
NextWeight	awpi	1wpi	1wpi	1wpi	1wpi
PheromoneUpdate	BO	ND	BO	BO	BO
N^{upd}	1	1	2	2	2
N^{ants}	10	10	5	10	10
MultiColonyWeights	overlap	overlap	overlap	overlap	overlap
MultiColonyUpdate	region	region	region	region	region

Figure 10 Konfigurasi dengan komponen algoritma MOACO (sumber: Manuel Lopez, 2012)

Manuel Lopez menerapkan 10 konfigurasi ini untuk contoh uji, dan melakukan 15 berjalan independen satu sama konfigurasi dengan biji acak yang berbeda. Dan membuat perbandingan, dengan mengevaluasi algoritma MOACO dari literature. Manuel Lopez juga mengevaluasi kualitas set nondominated ditemukan di tes ini berjalan dengan cara baik hypervolume dan ukuran epsilon. Hasilnya mengarah pada kesimpulan yang sama secara independen dari ukuran kualitas yang digunakan dalam analisis. Selain itu, tidak ada perbedaan yang signifikan menurut uji tanda, antara konfigurasi saat ditemukan menggunakan hypervolume sebagai kriteria evaluasi I / F-Ras dan yang ditemukan saat

menggunakan ukuran epsilon. Dengan demikian metode ini mampu menemukan desain baru yang melampaui-the-art-of arus dalam literatur MOACO. Tidak mengherankan, desain baru ini mirip dengan varian multi-koloni BicriterionAnt. Namun, itu termasuk perbedaan yang signifikan, seperti penggunaan pembaruan terbaik-ofobjective, nilai kecil N^{upd}, lebih dari satu semut per weight, dan satu-weight-per-iterasi (1wpi) [21].

b) Konfigurasi underlying ACO Algorithm

Dalam percobaan berikutnya, Manuel Lopez [21] secara sembarangan memilih salah satu konfigurasi yang ditemukan ketika mengkonfigurasi komponen framework multi-objective MOACO, melaksanakan konfigurasi otomatis dari pengaturan dari algoritma ACO yang mendasari. Domain pengaturan algoritma ACO dijelaskan pada Tabel VII. Bukannya mengkonfigurasi langsung jumlah semut per koloni N^a, mengkonfigurasi parameter a_f pengganti yang mendefinisikan jumlah semut dalam ketergantungan dari ukuran contoh n, yaitu, $N^a = 6 \cdot a_f \cdot [n / 100]$. Kemudian kalikan faktor ini dengan enam agar dapat menentukan $N^{weights}$ baik sebagai $N^a / 3$ atau $N^a / 2$ dalam percobaan kemudian [21].

Manuel Lopez melakukan lima kali pengulangan independen dari proses konfigurasi menggunakan hypervolume sebagai kriteria evaluasi, dan setiap pengulangan dihentikan setelah 1000 berjalan dari framework MOACO. Sebagai perbandingan, digunakan juga mengkonfigurasi cara pengaturan algoritma ACO dari versi multi-koloni BicriterionAnt, yang ditemukan menjadi algoritma MOACO terbaik dari literatur di bagian sebelumnya [21].

Component	Domain
a_f	$\{1, 2, \ldots, 25\}$ where $N^3 = 6 \cdot a_f \cdot \lfloor n/100 \rfloor$
ρ	$[0.01, 0.99]$
q_0	$[0.25, 0.99]$ or $q_0 = 0$
α	$[0, 5]$
β	$[0, 5]$

Figure 11 Domain parameter setting pada algoritma ACO (MMAS) (sumber: Manuel Lopez, 2012)

Muncul pertanyaan apakah tidak akan lebih baik untuk mengkonfigurasi kedua komponen multi-tujuan dan pengaturan parameter ACO dari framework MOACO sekaligus, bukannya berturut-turut seperti yang dilakukan di atas. Untuk menjawab pertanyaan ini, kita melaksanakan konfigurasi otomatis dari semua parameter dari kerangka MOACO sekaligus, dengan domain yang dijelaskan dalam Tabel IV dan VII, dan dengan budget 2000 percobaan, yaitu setara dengan usaha gabungan dihabiskan di dua-tahap proses konfigurasi otomatis sebelumnya. Konfigurasi yang dihasilkan diberikan sebagai bahan tambahan. Hasil yang disajikan di bawah ini menunjukkan bahwa ada perbedaan yang signifikan dalam mendukung secara otomatis mengkonfigurasi semua parameter sekaligus.

Untuk perbandingan konfigurasi yang ditemukan di berbagai prosedur konfigurasi otomatis, dilakukan 15 berjalan independen satu sama konfigurasi pada contoh uji. Manuel Lopez mengevaluasi hasil dengan menggunakan ukuran hypervolume, dan sesuai boxplots. Plot menunjukkan bahwa secara otomatis mengkonfigurasi pengaturan algoritma ACO dari BicriterionAnt sangat meningkatkan kinerja, dan untuk kasus dengan 500 kota, itu melampaui kualitas konfigurasi MOACO dengan pengaturan algoritma standar ACO. Meskipun demikian, dengan secara otomatis mengkonfigurasi pengaturan dari algoritma ACO yang mendasari framework MOACO, peningkatan hypervolume bahkan lebih luar

biasa, jelas mengalahkan konfigurasi terbaik dari BicriterionAnt. Plot tidak menunjukkan perbedaan yang jelas antara otomatis mengkonfigurasi framework MOACO menggunakan pendekatan dua tahap atau dengan mengkonfigurasi semua parameter sekaligus. Namun, konfigurasi diperoleh dengan pendekatan dua tahap mendapatkan hypervolume lebih rendah, yaitu lebih buruk, daripada sepenuhnya dikonfigurasi dalam 80% dari berjalan. Tanda-test menegaskan bahwa perbedaan ini secara statistik signifikan [21].

4. MOACO Framework dan Local Search

Penelitian yang dilakukan Manuel Lopez [21] memberikan hasil yang disajikan di atas secara meyakinkan menunjukkan bahwa konfigurasi otomatis mengarah ke desain yang lebih baik dari algoritma MOACO daripada yang disajikan dalam literatur. Meskipun sejumlah besar pekerjaan pada algoritma MOACO tidak mempertimbangkan penggunaan pencarian lokal, diketahui bahwa, pada algoritma BTSP, MOACO dengan pencarian lokal jauh mengungguli tanpa pencarian lokal. Oleh karena itu, Manuel Lopez mengulangi konfigurasi otomatis hibridisasi framework MOACO dan pencarian lokal [21].

a) Experimental setup

Pencarian lokal yang digunakan di sini adalah perbaikan berulang algoritma didasarkan pada lingkungan 2-exchange (2-opt). Pencarian lokal diterapkan untuk setiap solusi yang dibangun oleh semut. The BTSP dikonversi menjadi TSP single-objective dengan cara jumlah agregasi tertimbang dari dua matriks jarak. Jika semut menggunakan vektor bobot untuk membangun solusi, pencarian lokal selanjutnya akan menggunakan vektor bobot yang sama. Jika tidak, jika semut tidak menggunakan beban untuk konstruksi solusi, weight masih ditugaskan untuk setiap semut berikut pengaturannya tetapi mereka hanya digunakan oleh pencarian lokal. Pencarian lokal juga memanfaatkan teknik speed-up

standar untuk TSP, misalnya daftar memerintahkan tepi calon ukuran 20 dihitung untuk setiap vektor berat. Semut menggunakan daftar calon yang berbeda dari ukuran 20 untuk konstruksi, yang diperoleh dengan menyortir tepi sesuai dengan dominasi Peringkat [21].

Parameter sisa framework MOACO sama dengan pengaturan sebelumnya tanpa pencarian lokal (Tabel III, IV, dan VII) kecuali bahwa: (i) nilai default ρ adalah 0,2; (ii) jumlah semut tidak tergantung pada ukuran contoh dan sekarang dihitung sebagai $N^a = 6 \cdot a_ß$; dan (iii) batas waktu masing-masing berjalan sekarang $4 \cdot (n / 100)^2$. Sebagai contoh pelatihan, kami menghasilkan 10 BTSP seragam acak contoh Euclidean untuk setiap n = {500, 600, 700, 800, 1000} node (60 kasus total). Perbandingan dan evaluasi konfigurasi diperoleh menggunakan satu set yang berbeda dari 15 contoh uji, 3 contoh setiap ukuran n [21].

b) Automatic configuration MOACO+ls

Manuel Lopez [21] melaksanakan empat setup konfigurasi otomatis yang terpisah dari MOACO + ls kerangka kerja untuk BTSP sebagai berikut:

1) Melakukan konfigurasi komponen multi-objective framework MOACO dan menggunakan pengaturan default untuk algoritma ACO yang mendasarinya.
2) Melakukan konfigurasi pengaturan dari algoritma ACO yang mendasari dan menggunakan salah satu desain MOACO ditemukan pada langkah sebelumnya.
3) Melakukan konfigurasi semua pengaturan dari framework MOACO sekaligus menggunakan dua kali budget eksperimen seperti untuk dua langkah sebelumnya. Tujuan dari konfigurasi ini adalah untuk menyelidiki apakah

pendekatan konfigurasi dua tahap lebih atau kurang efektif daripada mengkonfigurasi semua parameter sekaligus.

4) Melakukan konfigurasi pengaturan algoritma ACO dari BicriterionAnt + ls untuk perbandingan dengan hasil lainnya.

Manuel Lopez melakukan lima kali pengulangan setiap pengaturan konfigurasi otomatis, dan budget dari setiap pengulangan diatur ke 1000 percobaan (2000 untuk mengkonfigurasi semua pengaturan sekaligus). Alat konfigurasi menggunakan ukuran hypervolume sebagai kriteria evaluasi. Hal ini cukup untuk mengatakan di sini bahwa konfigurasi menunjukkan variabilitas yang lebih besar dari pengaturan, dijelaskan oleh fakta bahwa pencarian lokal mengurangi efek parameter lainnya.

Konfigurasi ditemukan secara otomatis dievaluasi pada contoh uji dengan melakukan 15 berjalan independen satu sama konfigurasi, dan menghitung ukuran hypervolume set nondominated yang dihasilkan. Hasil yang didapatkan mengikuti kesimpulan yang diuraikan dalam bagian sebelumnya. Pertama, kinerja BicriterionAnt sangat meningkat dengan mengkonfigurasi pengaturan algoritma ACO nya. Kedua, meskipun peningkatan ini, secara otomatis dikonfigurasi BicriterionAnt tidak mengungguli desain otomatis menemukan kerangka MOACO dengan pengaturan algoritma standar ACO. Selain itu, secara otomatis mengkonfigurasi pengaturan algoritma ACO dari suatu desain yang baik mengarah untuk lebih meningkatkan kualitas, jelas mengalahkan hasil terbaik dari BicriterionAnt. Akhirnya, meskipun boxplots tidak secara visual menunjukkan

perbedaan yang jelas antara proses konfigurasi dua tahap vs sepenuhnya mengkonfigurasi MOACO + ls kerangka sekaligus, tanda-test menunjukkan bahwa ada probabilitas signifikan lebih besar, sekitar 0,84, untuk memperoleh hypervolume lebih besar dengan konfigurasi diperoleh dengan metode yang terakhir. Hal ini sesuai dengan kesimpulan yang dicapai saat otomatis mengkonfigurasi kerangka MOACO tanpa pencarian lokal [21].

Hasil ini jelas menunjukkan bahwa telah ditemukan desain baru dari algoritma MOACO yang mengungguli keadaan sebelumnya seni dalam literatur MOACO untuk BTSP. Lebih penting lagi, Manuel Lopez telah menemukan desain ini secara semi-otomatis, di mana sebagian besar upaya telah dihabiskan pada definisi pilihan desain alternatif, meninggalkan tugas mencari desain yang benar untuk karenanya, berisi alat-alat otomatis dan. Fakta bahwa beberapa pilihan desain tidak pernah muncul dalam beberapa konfigurasi memberikan indikasi yang jelas bahwa mereka tidak berkontribusi terhadap kinerja algoritma. Di sisi lain, variabilitas desain menunjukkan bahwa ada beberapa alternatif desain yang menghasilkan kualitas yang sama dari hasil [21].

Dengan demikian, penelitian ini akan menggunakan framework yang ditemukan oleh Manuel Lopez. Penelitian ini akan mencoba melakukan implementasi framework ini untuk memecahkan masalah pada objek mitigasi bencana.

3.3. Mitigasi Bencana

Mitigasi bencana merupakan manajemen bencana secara umum. Mitigitasi memiliki tiga bagian yaitu, proses pendeteksian (*detection)*, proses penanganan saat terjadi bencana (*respons)*, dan penanganan pasca bencana (*recovery*) [24] [25]. Pendektesian membahas menyangkut bagaimana secara dini bencana dalam dapat dideteksi sehingga dapat membangun *awardness* dan kesiap-siagaan dalam menghadapi bencana alam [26]. System deteksi ini biasa dikenal dengan istilah *early warning system* [27] [24].

Sedangkan respon adalah bagian dimana bencana dihadapi dan dilakukan penganggulangan. Kegiatan yang dilakukan seperi melakukan evakuasi dan penyelamatan korban bencana melalui jalur evakuasi yang dipilih. Cara menentukan jalur evakuasi haruslah dipilih jalur paling cepat untuk dilalui [28]. Salah satu cara yang dapat digunakan untuk menentukan jalur evakuasi adalah dengan menggunakan algoritma komputasi seperti algoritma ant colony [11] [29].

Proses terakhir dalam mitigasi bencana adalah recovery. Proses ini adalah proses pemulihan pasca bencana alam seperti bagaimana penyaluran logistic ke shelter pengungsian dimana para korban bencana ditangani sementara [30] [31]. Pada proses ini, dapat dilakukan penentuan shelter pengungsi secara cepat dan dapat dijangkau dalam waktu yang cepat [32] [33]. Factor lain yang harus dipertimbangkan adalah pemilihan jalur distribusi pengiriman bantuan yang dapat dikirim secara cepat dan tepat di shelter pengungsian [33] [34] [35].

Menurut Iman Satyarno (2016), tujuan dari mitigasi adalah untuk mengurangi jumlah korban jiwa dan mengurangi kerugian harta benda [36]. Manajemen kebencanaan dapat dibagi menjadi (dua) bagian besar yaitu sebelum *(berofe)* dan Sesudah *(after)* bencana terjadi. Pada bagian *before* kegiatan yang dapat dilakukan adalah mitigasi dan *reduction* sedangkan bagian *after* adalah kegiatan untuk memberi respond an melakukan rehabilitasi dan *reconstruction* [36]. Pada penelitian ini focus utama yang dipelajari adalah proses pada bagian mitigasi bencana saja. Bagian lainnya dapat dibahas secara terpisah.