SKRIPSI

OPTIMASI MULTIOBJEKTIF PENEMPATAN MESIN VIRTUAL DAN PENENTUAN RUTE JARINGAN PADA *CLOUD DATA CENTER* BERBASIS ALGORITMA GENETIKA *NONDOMINATED SORTING*

MULTI-OBJECTIVE OPTIMIZATION OF VIRTUAL MACHINE PLACEMENT AND NETWORK ROUTING IN CLOUD DATA CENTERS BASED ON NONDOMINATED SORTING GENETIC ALGORITHM

GUSTI AGUNG RAMA AYUDHYA
20/459266/PA/19927

PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2025

SKRIPSI

OPTIMASI MULTIOBJEKTIF PENEMPATAN MESIN VIRTUAL DAN PENENTUAN RUTE JARINGAN PADA *CLOUD DATA CENTER* BERBASIS ALGORITMA GENETIKA *NONDOMINATED SORTING*

MULTI-OBJECTIVE OPTIMIZATION OF VIRTUAL MACHINE PLACEMENT AND NETWORK ROUTING IN CLOUD DATA CENTERS BASED ON NONDOMINATED SORTING GENETIC ALGORITHM

Usulan Penelitian

GUSTI AGUNG RAMA AYUDHYA
20/459266/PA/19927

PROGRAM STUDI ILMU KOMPUTER
DEPERTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2025

HALAMAN PENGESAHAN

SKRIPSI

OPTIMASI MULTIOBJEKTIF PENEMPATAN MESIN VIRTUAL DAN PENENTUAN RUTE JARINGAN PADA *CLOUD DATA CENTER* BERBASIS ALGORITMA GENETIKA *NONDOMINATED SORTING*

Telah dipersiapkan dan disusun oleh

GUSTI AGUNG RAMA AYUDHYA
20/459266/PA/19927

Telah dipertahankan di depan Tim Penguji
pada tanggal Mei 2025

Susunan Tim Penguji

Drs. Medi, S.Kom., M.Cs	-
Pembimbing	Ketua Penguji
	-
	Anggota Penguji

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, Mei 2025

Gusti Agung Rama Ayudhya

Karya ini ku persembahkan kepada
Ibu, Bapak, dan adik-adikku tercinta serta teman-teman
seperjuangan di Ilmu Komputer Universitas Gadjah Mada

PRAKATA

Puji syukur penulis panjatkan ke hadapan Ida Hyang Widhi Wasa, Tuhan Yang Maha Esa, karena atas *asung kerta wara nugraha*-Nya, penulis dapat menyelesaikan skripsi ini dengan judul "Optimasi Multiobjektif Penempatan Mesin Virtual dan Penentuan Rute Jaringan pada *Cloud Data Center* Berbasis Algoritma Genetika *Nondominated Sorting*". Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari dukungan, bimbingan, dan bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Ayah dan Ibu, yang telah membimbing, mendukung, mendoakan, serta membiayai penulis hingga dapat menempuh pendidikan di Universitas Gadjah Mada.
2. Gusti Ayu Bulan Adhistanaya dan Gusti Agung Deva Maheswara, kedua adik penulis, serta seluruh keluarga besar yang senantiasa memberikan semangat dan dukungan selama proses penyusunan skripsi ini.
3. Bapak Drs. Medi, M.Kom., selaku Dosen Pembimbing, yang telah dengan sabar membimbing, memberikan ilmu, arahan, masukan, serta koreksi selama proses penulisan skripsi ini.
4. Almarhumah Ibu Anny Kartika Sari, S.Si., M.Sc. dan Bapak Lukman Heryawan, S.T., M.T., selaku Dosen Pembimbing Akademik, atas segala bimbingan, nasihat, dan bantuan selama penulis menempuh studi di Program Studi Ilmu Komputer.
5. Tim Penguji, yang telah memberikan kritik, saran, serta masukan berharga untuk penyempurnaan skripsi ini.
6. Seluruh dosen dan staf Fakultas MIPA UGM, khususnya di Program Studi Ilmu Komputer, yang telah memberikan ilmu dan dukungan selama proses studi.
7. Teman-teman Ilmu Komputer Angkatan 2020 dan OmahTI, yang telah menjadi rekan seperjuangan dan selalu siap membantu, baik dalam perkuliahan maupun dalam penyusunan skripsi.

8. Rekan-rekan Tim KKN Punung Periode 4 Tahun 2023 Unit JI-081 (Senandung Punung), dan warga Desa Wareng, Kabupaten Pacitan, Jawa Timur, atas kebersamaan, pengalaman, dan kisah tak terlupakan selama lima puluh hari pengabdian.
9. Seluruh staf Perpustakaan dan Arsip Universitas Gadjah Mada, yang telah menyediakan fasilitas dan referensi penting dalam mendukung penyusunan skripsi ini.
10. Seluruh pihak yang tidak dapat disebutkan satu per satu, yang turut membantu dan mendukung penulis selama proses penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis terbuka atas segala kritik dan saran yang membangun demi perbaikan ke depan. Penulis berharap skripsi ini dapat memberikan manfaat, baik dalam pengembangan ilmu komputer maupun sebagai referensi bagi penelitian selanjutnya.

Yogkarta, Mei 2025

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Optimasi Multiobjektif Penempatan Mesin Virtual dan Penentuan Rute Jaringan pada *Cloud Data Center* Berbasis Algoritma Genetika *Nondominated Sorting*

Oleh

Gusti Agung Rama Ayudhya
20/459266/PA/19927

Komputasi awan telah menjadi infrastruktur utama dalam teknologi informasi karena kemampuannya menyediakan sumber daya secara *scalable* dan elastis sesuai permintaan pengguna. Kemampuan ini didukung oleh virtualisasi, yang memungkinkan penyewaan sumber daya tanpa pengelolaan langsung oleh pengguna, sekaligus meningkatkan efisiensi dan keberlanjutan *data center*. Salah satu tantangan utama dalam komputasi awan adalah penempatan VM (*virtual machine* atau mesin virtual) pada PM (*physical machine* atau mesin fisik) dan rekayasa lalu lintas (*traffic engineering*) jaringan *data center*, yang harus mempertimbangkan berbagai efisiensi operasional, seperti konsumsi energi dan efisiensi sumber daya, serta *Quality of Service* (QoS), seperti alokasi *bandwidth* dan *latency* komunikasi antar-VM.

Optimasi penempatan VM dan penentuan rute menjadi semakin kompleks karena adanya kendala yang sering kali saling bertentangan, sehingga lebih cocok diselesaikan menggunakan metode optimasi multiobjektif. Masalah ini bersifat *NP-complete*, sehingga lebih cocok diselesaikan menggunakan algoritma heuristik dibandingkan algoritma eksak. Penelitian ini menggunakan NSGA-III (*Nondominated Sorting Genetic Algorithm* III), sebuah algoritma genetika untuk menyelesaikan masalah multiobjektif, dengan tujuan mengoptimalkan konsumsi energi, efisiensi sumber daya dalam penempatan VM, alokasi *bandwidth*, dan *latency* komunikasi dalam penentuan rute jaringan secara bersamaan.

Evaluasi performa NSGA-III dilakukan melalui simulasi menggunakan CloudSim Plus, dengan mempertimbangkan berbagai skenario seperti topologi jaringan, kebutuhan sumber daya VM, kapasitas PM, serta parameter NSGA-III. Performa NSGA-III akan diukur menggunakan beberapa metrik, termasuk *hypervolume* dan rata-rata jarak antargenerasi. Selain itu, solusi yang diperoleh NSGA-III akan dibandingkan dengan solusi eksak yang diperoleh *LP solver* (pemecah pemrograman linier) serta beberapa kombinasi algoritma heuristik untuk penempatan VM dan penentuan rute.

Kata Kunci: Komputasi awan, *data center*, penempatan mesin virtual,

konsumsi energi, efisiensi sumber daya, masalah *bin packing*, perutean jaringan, alokasi *bandwidth*, *latency*, masalah *multicommodity flow*, metode optimasi multiobjektif, *Nondominated Sorting Genetic Algorithm III* (NSGA-III)

ABSTRACT

Multi-Objective Optimization of Virtual Machine Placement and Network Routing in Cloud Data Centers Based on Nondominated Sorting Genetic Algorithm

By

Gusti Agung Rama Ayudhya
20/459266/PA/19927

Cloud computing has become a key infrastructure in information technology due to its ability to provide scalable and elastic resources on demand. This capability is supported by virtualization, which enables resource rental without direct management by users while also improving efficiency and sustainability of data centers. One of the main challenges in cloud computing is the placement of virtual machines (VMs) on physical machines (PMs) and traffic engineering in data center networks, which must consider various operational efficiency factors, such as energy consumption and resource utilization, as well as Quality of Service (QoS) aspects, such as bandwidth allocation and communication latency between VMs.

VM placement and network routing optimization becomes increasingly complex due to conflicting constraints, making multiobjective optimization methods suitable for solving this problem. On the top of that, this problem is NP-complete, making heuristic algorithms more suitable than exact algorithms. This study employs NSGA-III (Nondominated Sorting Genetic Algorithm III), a genetic algorithm for solving multi-objective problems, aiming to optimize energy consumption, resource efficiency in VM placement, bandwidth allocation, and communication latency in network routing simultaneously.

The performance of NSGA-III is evaluated through simulations using CloudSim Plus, considering various scenarios such as network topology, VM resource demands, PM capacity, and NSGA-III parameters. The performance of NSGA-III is measured using several metrics, including hypervolume and the average intergenerational distance. Additionally, the solutions obtained by NSGA-III are compared with exact solutions obtained from linear programming (LP) solvers and several heuristic algorithm combinations for VM placement and routing.

Keywords: cloud computing, data center, virtual machine placement, energy consumption, resource wastage, bin packing problem, network routing, bandwidth allocation, latency, multicommodity flow problem, multiobjective optimization method, Nondominated Sorting Genetic Algorithm

BAB I

PENDAHULUAN

1.1 Latar Belakang

Seiring perkembangan teknologi informasi, *cloud computing* (komputasi awan) telah menjadi infrastruktur utama bagi penyedia layanan komputasi. Komputasi awan memungkinkan penggunaan sumber dayaseperti penyimpanan, jaringan, daya komputasi, basis data, *platform*, dan layanan aplikasisecara efisien melalui Internet. Model komputasi ini semakin diminati karena mampu menyediakan sumber daya yang bersifat *scalable* dan elastis sesuai permintaan pengguna (*on-demand*).

Kemampuan ini dimungkinkan oleh teknologi virtualisasi. Virtualisasi memungkinkan pengguna menyewa dan mengakses sumber daya yang sudah diabstraksi tanpa perlu memiliki, mengelola, atau merawatnya secara langsung. Virtualisasi juga mendukung pendekatan *multi-tenancy*, yaitu penggunaan infrastruktur yang sama oleh banyak pengguna, sehingga penyedia layanan dapat menghemat biaya operasional (Cloudflare, t.t.). Dengan adanya virtualisasi, VM (*virtual machine* atau mesin virtual) dapat dibuat sebagai emulasi perangkat keras fisik yang mampu menjalankan sistem operasi dan aplikasi layaknya komputer fisik. VM ini dapat dengan mudah dijalankan, diskalakan, dimigrasikan, direplikasikan, atau dihancurkan, sehingga meningkatkan skalabilitas, elastisitas, fault-tolerance, ketersediaan, penghindaran bencana (*disaster avoidance*), dan pemulihan bencana (*disaster recovery*) (Hill dkk., 2013).

Namun, supaya dapat digunakan, VM harus ditempatkan pada PM (*physical machine* atau mesin fisik). Penempatan VM menjadi tantangan utama bagi penyedia layanan komputasi awan karena harus memperhatikan kebutuhan sumber daya VM serta kapasitas PM. Penempatan yang optimal dapat meningkatkan efisiensi data center dan metrik performa, seperti minimalisasi konsumsi energi, efisiensi penggunaan sumber daya, serta minimalisasi biaya komunikasi antar-VM.

Komunikasi antar-VM dalam *cloud data center* juga memainkan peran penting dalam kinerja aplikasi. *Cloud data center* skala besar, yang biasanya memiliki ribuan PM terhubung oleh perangkat jaringan, sering kali digunakan untuk mendukung aplikasi yang terdiri atas beberapa komponen saling bergantung.

Komponen yang saling berkomunikasi melalui jaringan dapat menyebabkan *delay*, yang berdampak pada jumlah tugas (*task*) yang dapat diproses oleh VM per satuan waktu. Penempatan VM dan pemilihan jalur komunikasi yang ideal mampu mengurangi *delay* komunikasi dengan menempatkan VM yang saling berkomunikasi intensif pada PM yang sama atau PM yang terhubung oleh jalur jaringan sesingkat mungkin.

Selain itu, manajemen energi di data center menjadi isu krusial dalam komputasi awan. Konsumsi energi tidak hanya memengaruhi biaya operasional tetapi juga berdampak pada lingkungan, seperti emisi karbon yang dihasilkan oleh perangkat fisik, terutama server. Menurut survei yang dilakukan oleh Lawrence Berkeley National Laboratory (2024), sekitar 60% total konsumsi energi di data center berasal dari mesin fisik. Mesin fisik *idle* (menyala tetapi tidak menjalankan *task* apapun) rata-rata mengonsumsi energi sebesar 70% dari energi yang digunakan oleh mesin fisik dengan utilisasi CPU maksimal (Beloglazov, Abawajy, dan Buyya, 2012). Dengan penempatan VM yang ideal, lebih banyak VM dapat ditempatkan pada PM dengan tingkat utilisasi tinggi. Mesin fisik yang *idle* dapat dimatikan, sehingga jumlah perangkat aktif berkurang dan konsumsi energi *data center* dapat ditekan. Pendekatan ini tidak hanya meningkatkan efisiensi energi tetapi juga mendukung keberlanjutan lingkungan.

Kebutuhan mengoptimalkan beberapa metrik performa sekaligus: konsumsi energi *data center*, efisiensi penggunaan sumber daya komputasi, alokasi *bandwidth*, dan *latency* komunikasi antar-VM, membuat penempatan VM semakin kompleks. Pengoptimalan metrik-metrik tersebut sering kali saling bertentangan, di mana pengoptimalan salah satu metrik dapat memperburuk metrik lainnya. Selain itu, terdapat kendala yang harus dipenuhi, seperti kebutuhan sumber daya dan kapasitas PM, serta pemilihan jalur komunikasi antar-VM, yang membatasi ruang solusi. Untuk itu, setiap metrik yang dipertimbangkan harus dioptimalkan secara setara tanpa memberikan prioritas pada salah satu metrik. Masalah ini sangat cocok untuk diselesaikan menggunakan metode optimasi multiobjektif.

Masalah penempatan VM merupakan salah satu bentuk masalah *bin packing* (Fatima dkk., 2018), sedangkan masalah pemilihan jalur komunikasi antar-VM merupakan salah satu bentuk masalah *multicommodity flow* (Fortz, Gouveia & Joyce-Moniz, 2017). Keduanya dikenal memiliki kompleksitas *NP-complete*, sehingga secara keseluruhan, masalah ini juga berkompleksitas *NP-complete*. Oleh karena itu, masalah ini kurang cocok diselesaikan dengan algoritma eksak. Sebagai gantinya,

berbagai algoritma heuristik dan metaheuristik telah dikembangkan untuk mencari solusi yang cukup optimal dengan pendekatan yang lebih efisien,

Dalam penelitian ini, penulis menggunakan algoritma NSGA-III (*Nondominated Sorting Genetic Algorithm III*), salah satu varian algoritma genetika yang dirancang untuk menyelesaikan masalah multiobjektif. Algoritma ini akan digunakan untuk menempatkan VM pada PM serta memilih rute komunikasi antar-VM, dengan tujuan mengoptimalkan empat metrik metrik performa sekaligus: konsumsi energi *cloud data center*, efisiensi penggunaan sumber daya komputasi, alokasi *bandwidth*, dan *latency* komunikasi antar-VM.

Kinerja NSGA-III akan dievaluasi pada berbagai skenario, mencakup parameter NSGA-III itu sendiri, variasi topologi jaringan, kebutuhan sumber daya setiap VM, kapasitas sumber daya tiap PM, *bandwidth* yang diperlukan oleh pasangan VM yang saling berkomunikasi, serta kapasitas dan *latency* setiap *link* dalam jaringan. Penelitian ini menggunakan CloudSim Plus untuk memodelkan dan mensimulasikan lingkungan komputasi awan beserta algoritma penempatan VM dan pemilihan jalur komunikasinya.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, terdapat beberapa masalah yang hendak dipecahkan melalui penelitian ini:

1. Bagaimana mengembangkan algoritma penempatan VM dan penentuan rute komunikasi antar-VM yang mampu menangani *trade-off* antara konsumsi energi, efisiensi penggunaan sumber daya, alokasi *bandwidth*, dan *latency* komunikasi antar-VM secara bersamaan?
2. Bagaimana memastikan algoritma yang dikembangkan dapat menghasilkan solusi optimal Pareto yang menggambarkan berbagai skenario kompromi antar metrik performa, sehingga memungkinkan pembuat keputusan memilih solusi yang paling sesuai?
3. Bagaimana kinerja algoritma NSGA-III dalam menyelesaikan masalah penempatan VM dan penentuan jalur komunikasi antar-VM, seperti kualitas *Pareto front*, efisiensi komputasi, dan stabilitas solusi dalam berbagai skenario lingkungan *data center*?

1.3 Batasan Masalah

Supaya penelitian lebih terfokus, terdapat beberapa batasan yang diterapkan:

1. **Evaluasi melalui Simulasi**
   Kinerja algoritma dievaluasi melalui simulasi menggunakan CloudSim Plus. Oleh karena itu, performa algoritma dalam lingkungan nyata yang lebih dinamis dapat berbeda dengan hasil simulasi.
2. **Asumsi Lingkungan Statis**
   Penelitian ini mengasumsikan lingkungan *cloud* bersifat statis, di mana semua parameter permasalahan, seperti jumlah VM dan PM, kebutuhan sumber daya dari VM, kapasitas PM, serta karakteristik jaringan, tidak mengalami perubahan selama simulasi berlangsung. Dengan demikian, algoritma yang dikembangkan bersifat *offline* dan tidak menangani perubahan dinamis, seperti kedatangan atau penghentian VM secara *real-time*.
3. **Komunikasi Antar-VM dalam PM yang Sama**
   Jika dua VM ditempatkan pada PM yang sama, komunikasi antar-VM dilakukan melalui *memory sharing* tanpa memanfaatkan jaringan tambahan. Kebutuhan memori yang dialokasikan untuk setiap VM diasumsikan sudah mencakup kebutuhan tambahan untuk *memory sharing*.
4. **Penggunaan *Software Defined Network* (SDN) pada *Data Center***
   Penelitian ini mengasumsikan bahwa informasi mengenai topologi jaringan *data center* tersedia bagi pembuat keputusan, termasuk struktur topologi, kapasitas setiap *link*, dan kecepatan transmisinya. Dengan adanya SDN, jalur komunikasi antar-VM dapat ditentukan berdasarkan informasi tersebut.
5. **Penggunaan Banyak Jalur untuk Setiap Komunikasi**
   Setiap komunikasi antar-VM dapat menggunakan lebih dari satu jalur secara bersamaan (*multipath routing*). Hal ini didukung oleh protokol seperti *Multipath Transmission Control Protocol* (MP-TCP) yang memungkinkan distribusi lalu lintas melalui beberapa jalur dalam jaringan.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk menyelesaikan masalah yang telah dirumuskan, antara lain:

1. Mengembangkan algoritma penempatan VM dan penentuan rute komunikasi berbasis NSGA-III yang mampu menangani *trade-off* antara empat metrik: konsumsi energi, penggunaan sumber daya, alokasi *bandwidth*, dan *latency* komunikasi antar-VM.
2. Merancang algoritma yang menghasilkan himpunan solusi optimal Pareto, sehingga pembuat keputusan dapat memilih solusi yang sesuai dengan kebutuhan berdasarkan kompromi antar metrik.
3. Mengevaluasi kinerja algoritma NSGA-III melalui simulasi menggunakan CloudSim Plus dengan mengukur kualitas *Pareto front*, efisiensi komputasi, dan stabilitas solusi pada berbagai skenario lingkungan *data center*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Kontribusi terhadap pengembangan algoritma penempatan VM dan pemilihan rute komunikasi antar-VM melalui pendekatan optimasi multiobjektif berbasis NSGA-III. Hasil penelitian ini dapat menjadi acuan bagi penelitian selanjutnya dalam bidang penjadwalan dan pengelolaan sumber daya di *cloud computing*.
2. Potensi penerapan dalam lingkungan komputasi awan nyata, khususnya dalam meningkatkan efisiensi penggunaan sumber daya, mengoptimalkan konsumsi energi, dan mengurangi *latency* komunikasi antar-VM.
3. Menyediakan referensi bagi penyedia layanan komputasi awan dalam merancang strategi penempatan VM yang mempertimbangkan kompromi antara berbagai metrik performa, sehingga dapat meningkatkan QoS dan efisiensi operasional *data center*.

1.6 Metodologi Penelitian

Penelitian ini dilakukan melalui beberapa tahapan sebagai berikut:

1. **Studi Literatur**
   Tahap ini mencakup kajian terhadap penelitian terdahulu yang berkaitan dengan penempatan VM, pemilihan rute komunikasi antar-VM, serta metode optimasi multiobjektif berbasis algoritma evolusioner. Literatur yang dikaji mencakup

buku, jurnal ilmiah, serta publikasi konferensi yang relevan. Selain itu, studi juga dilakukan terhadap teknologi yang digunakan, seperti NSGA-III sebagai algoritma optimasi, CloudSim Plus untuk simulasi, dan DOCPLEX sebagai *LP solver*.

2. **Perancangan Algoritma Penempatan VM dan Lingkungan Simulasi**
   Pada tahap ini, dilakukan perancangan model optimasi multiobjektif yang mencakup:
   (a) Formulasi masalah optimasi dalam bentuk fungsi objektif dan kendala.
   (b) Pemilihan metode representasi kromosom untuk solusi penempatan VM dan pemilihan rute komunikasi.
   (c) Perancangan operator algoritma genetika untuk NSGA-III, termasuk crossover, mutasi, dan mekanisme perbaikan.
   (d) Penentuan skenario simulasi, seperti topologi jaringan, konfigurasi VM dan PM, serta parameter simulasi lainnya.
3. **Implementasi Algoritma**
   Tahap ini mencakup implementasi algoritma NSGAIII dalam lingkungan simulasi CloudSim Plus. Implementasi meliputi:
   (a) Implementasi algoritma NSGA-III ke dalam program Java.
   (b) Integrasi algoritma NSGA-III dengan CloudSim Plus untuk menilai performa solusi yang dihasilkan.
   (c) Pengujian awal untuk memastikan algoritma berjalan sesuai dengan perancangan.
4. **Pengujian Algoritma pada Lingkungan Simulasi dan Evaluasi Kinerja**
   Algoritma yang telah diimplementasikan diuji pada berbagai skenario simulasi dengan variasi parameter, seperti:
   (a) Topologi jaringan yang berbeda (misalnya, FatTree, BCube, DCell).
   (b) Jumlah VM dan PM yang bervariasi.
   (c) Kebutuhan sumber daya setiap VM dan kapasitas sumber daya di setiap PM yang bervariasi.

   Evaluasi kinerja dilakukan dengan membandingkan kualitas solusi Pareto yang dihasilkan, efisiensi komputasi algoritma, serta stabilitas hasil dalam berbagai skenario.

5. **Penulisan Laporan**
   Hasil penelitian yang mencakup formulasi masalah, perancangan algoritma, hasil eksperimen, dan analisis kinerja algoritma didokumentasikan dalam bentuk skripsi.

1.7 Sistematika Penulisan

Skripsi ini ditulis dalam tujuh bab dengan struktur sebagai berikut:

Bab I: Pendahuluan
Bab ini membahas latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan skripsi ini.

Bab II: Tinjauan Pustaka
Bab ini membahas penelitian terdahulu yang berkaitan dengan metode penempatan VM, optimasi multiobjektif, serta pemodelan *cloud data center*. Selain itu, tinjauan terhadap algoritma evolusioner seperti NSGA-III juga disertakan untuk memberikan dasar pengembangan metode dalam penelitian ini.

Bab III: Landasan Teori
Bab ini membahas teori dan konsep yang mendukung penelitian, termasuk komputasi awan, metode optimasi multiobjektif, algoritma genetika, dan NSGA-III. Selain itu, bab ini menguraikan cara kerja NSGA-III serta formulasi masalah penempatan VM dan pemilihan jalur komunikasi antar-VM sebagai masalah optimasi multiobjektif.

Bab IV: Analisis dan Rancangan Sistem
Bab ini membahas analisis masalah, perancangan algoritma, perancangan lingkungan simulasi, serta rancangan evaluasi kinerja algoritma.

Bab V: Implementasi
Bab ini membahas implementasi algoritma NSGA-III, integrasi NSGA-III dengan lingkungan simulasi CloudSim Plus, pengaturan parameter eksperimen, serta mengevaluasi kinerja algoritma berdasarkan skenario yang dipertimbangkan dalam simulasi.

Bab VI: Hasil dan Pembahasan
Bab ini membahas hasil evaluasi kinerja algoritma berdasarkan simulasi yang

telah dilakukan. Analisis dilakukan terhadap kualitas solusi Pareto, efisiensi komputasi, serta stabilitas algoritma dalam berbagai skenario simulasi.

Bab VII: Kesimpulan

Bab ini berisi kesimpulan penelitian serta saran untuk pengembangan lebih lanjut.

BAB II

TINJAUAN PUSTAKA

Seperti yang telah dibahas pada bab sebelumnya, masalah penempatan mesin virtual memiliki kompleksitas *NP-complete*. Oleh karena itu, berbagai algoritma heuristik dan metaheuristik telah dikembangkan untuk menangani masalah ini secara lebih efisien.

Metode-metode tersebut dirancang untuk mengoptimalkan berbagai metrik performa, terutama konsumsi daya server dan pemborosan sumber daya. Meskipun berbagai model konsumsi daya server (Ahmed, Bollen & Alvarez, 2021) dan model penggunaan sumber daya telah dikembangkan, pada sebagian besar penelitian, konsumsi daya server diukur berdasarkan model yang dikembangkan oleh Beloglazov, Abawajy, dan Buyya (2021), sementara pemborosan sumber daya dihitung menggunakan model dari Gao dkk. (2013). Sejumlah penelitian menyederhanakan pengukuran konsumsi energi *data center* dengan cara menghitung banyaknya server yang aktif (sedang menjalankan mesin virtual).

2.1 Algoritma Penempatan Mesin Virtual

2.1.1 Metode Heuristik

Beberapa adaptasi algoritma klasik untuk masalah *bin packing* seperti *First Fit* (FF), *First Fit Decreasing* (FFD), *Random Fit* (RF), *Best Fit* (BF), dan *Best Fit Decreasing* (BFD), dapat digunakan untuk menentukan penempatan VM yang optimal (Alharabe, Rakrouki & Aljohani, 2022). Namun, solusi yang didapat belum cukup optimal. Selain itu, algoritma ini lebih cocok digunakan untuk mengoptimalkan satu objektif saja, seperti konsumsi energi. Untuk memperoleh solusi yang lebih optimal, metode seperti MinPR (Azizi, Zandsalimi & Li, 2020), GRVMP (*Greedy Randomized Virtual Machine Placement*) (Azizi dkk., 2021), dan CRBFF (*Combinated Random Best First Fit*) (Yousefi & Babamir, 2024) dikembangkan untuk meminimalkan konsumsi energi sekaligus mengurangi pemborosan sumber daya. Selain itu, algoritma non-*greedy* seperti WPRVMP (*Weighted PageRank-based Virtual Machine Placement*) memanfaatkan algoritma *weighted PageRank* untuk mengurangi jumlah server aktif sambil memaksimalkan pemanfaatan sumber daya server tersebut.

2.1.2 Metode Metaheuristik

Pendekatan metaheuristik, khususnya algoritma evolusioner, banyak digunakan dalam optimasi penempatan mesin virtual. Gao dkk. (2013) mengembangkan VMPACS (*Virtual Machine Placement with Ant Colony System*) berbasis ACO (*Ant Colony Optimization*) untuk meminimalkan konsumsi daya server dan pemborosan sumber daya. Alharabe, Rakrouki, dan Aljohani (2022) memperkenalkan HACOS, yang mengintegrasikan ACO dengan *simulated annealing* untuk mengoptimalkan *network traffic* dan tingkat penggunaan maksimum pada *link* jaringan. Liu dkk. (2018) menciptakan OEMACS untuk mengurangi jumlah server aktif dalam *data center*. Wei dkk. (2019) mengembangkan AP-ACO (*Adaptive Parameter Ant Colony Optimization*), yang parameternya dapat beradaptasi, untuk meminimalkan konsumsi daya dan biaya komunikasi antar-VM.

ACO (*Ant Colony Optimization*) Pendekatan metaheuristik, khususnya algoritma evolusioner, banyak digunakan dalam optimasi penempatan mesin virtual. Gao dkk. (2013) mengembangkan VMPACS (*Virtual Machine Placement with Ant Colony System*) berbasis ACO (*Ant Colony Optimization*) untuk meminimalkan konsumsi daya server dan pemborosan sumber daya. Alharabe, Rakrouki, dan Aljohani (2022) memperkenalkan HACOS, yang mengintegrasikan ACO dengan *simulated annealing* untuk mengoptimalkan *network traffic* dan tingkat penggunaan maksimum pada *link* jaringan. Liu dkk. (2018) menciptakan OEMACS untuk mengurangi jumlah server aktif dalam *data center*. Wei dkk. (2019) mengembangkan AP-ACO (*Adaptive Parameter Ant Colony Optimization*), yang parameternya dapat beradaptasi, untuk meminimalkan konsumsi daya dan biaya komunikasi antar-VM.

Algoritma Genetika Algoritma genetika banyak digunakan dalam optimasi penempatan VM, termasuk dengan metode pengkodean yang terinspirasi dari masalah *bin packing*. Metode pengkodean standar merepresentasikan solusi sebagai *array* yang menunjukkan indeks kotak tempat setiap item diletakkan. Namun, Falkenauer (1992) mengusulkan *Grouping Genetic Algorithm* (GGA) untuk mengatasi kelemahan pengkodean standar dengan menambahkan daftar label yang diperbolehkan, sehingga *crossover* dan mutasi tetap mempertahankan struktur partisi.

Wu (2021) menerapkan GGA untuk menempatkan VM pada PM identik guna meminimalkan konsumsi energi. Xu & Fortes (2010) mengombinasikan GGA dengan logika *fuzzy* untuk meminimalkan pemborosan sumber daya, konsumsi daya, dan

suhu tertinggi PM, dengan menggunakan *hash table* sebagai representasi kromosom. Liu dkk. (2014) mengadaptasi GGA dalam *Nondominated Sorting Genetic Algorithm* untuk mengoptimalkan jumlah PM aktif, lalu lintas jaringan, dan keseimbangan penggunaan sumber daya, meskipun tanpa mempertimbangkan topologi jaringan secara eksplisit. Sonklin & Sonklin (2023) menerapkan GGA untuk penempatan VM berdasarkan tipe yang telah ditentukan penyedia layanan *cloud*.

Tang & Pan (2014) mengasumsikan topologi jaringan *data center* berbasis hierarki tiga tingkat: *core*, *aggregation*, dan *edge*. Mereka mengklasifikasikan komunikasi antar-VM ke dalam empat kategori berdasarkan lokasi VM, dengan tujuan meminimalkan konsumsi energi jaringan dan PM. Meskipun menggunakan algoritma genetika standar, mereka menerapkan algoritma khusus untuk memperbaiki kromosom yang rusak dan prosedur optimasi lokal guna mengurangi jumlah PM aktif.

Metode Metaheuristik Lainnya Selain ACO dan algorithm genetika, algoritma evolusioner lainnya juga diterapkan. Balaji, Kiran, dan Kumar (2023) menggunakan *firefly algorithm* untuk meminimalkan konsumsi daya, sementara Ghetas (2021) menerapkan *monarch butterfly optimization* dalam MBO-VM untuk mengoptimalkan konsumsi daya dan pemborosan sumber daya. Tripathi, Pathak, dan Vidyarthi (2020) memodifikasi BDA (*Binary Dragonfly Algorithm*) menjadi VMPDA (*Virtual Machine Placement using Dragonfly Algorithm*) untuk mengurangi pemborosan sumber daya. Zhao, Zhou, dan Li (2019) mengembangkan GATA, algoritma hibrida berbasis algoritma genetika dan *tabu search*, untuk meminimalkan konsumsi daya dan meningkatkan *load balance*.

2.1.3 Metode *Machine Learning*

Metode *machine learning*, khususnya *reinforcement learning* (RL), juga banyak digunakan. Caviglione (2021) memanfaatkan *deep reinforcement learning* untuk meminimalkan konsumsi daya server, risiko gangguan perangkat keras, dan interferensi antar-VM. Ghasemi, Haghighat, dan Keshavarzi (2024) mengembangkan dua algoritma untuk menentukan penempatan VM yang bertujuan meminimalkan penggunaan energi, mengurangi pemborosan sumber daya, dan memaksimalkan *load balance*: VMPMFuzzyORL dan MRRL. VMPMFuzzyORL mengintegrasikan *reinforcement learning* (RL) dengan sistem *fuzzy*, sementara MRRL menggabungkan RL dengan algoritma *k-means*. Pada MRRL, algoritma *k-means* digunakan untuk membentuk klaster-klaster VM, sedangkan RL memetakan setiap klaster ke server

tertentu. Sebaliknya, pada VMPMFuzzyORL, RL langsung digunakan untuk memetakan masing-masing VM ke server tertentu, dengan *reward* dari setiap aksi ditentukan oleh sistem *fuzzy* yang mengevaluasi ketiga metrik performa tersebut. Qin dkk. (2020) memperkenalkan VMPMORL untuk meminimalkan konsumsi energi dan pemborosan sumber daya. Pada VMPMORL, MDP (*Markov Decision Process*) dimodifikasi menjadi MDP multi-objektif di mana *reward signal* dan $\widehat{Q}$-*value* untuk setiap objektif direpresentasikan sebagai vektor *reward* dan vektor $\widehat{Q}$-*value*. Jarak setiap vektor $\widehat{Q}$-value terhadap titik utopia, titik dengan koordinat ke-i yang berupa nilai terbaik untuk fungsi objektif ke-i, dihitung menggunakan metrik Chebyshev. Aksi dengan vektor $\widehat{Q}$-value yang memiliki jarak terkecil dari titik utopia dicari menggunakan algoritma ϵ-*greedy*.

2.2 Metode Optimasi Multiobjektif untuk Masalah Penempatan Mesin Virtual

Beberapa metode yang disebutkan sebelumnya, seperti VMPMORL (Qin dkk, 2020) dan algoritma buatan Caviglione (2021), merumuskan masalah penempatan mesin virtual sebagai optimasi multiobjektif, di mana solusi diperoleh dengan menyeimbangkan setiap objektif (metrik performa) secara bersamaan untuk menghasilkan sejumlah solusi Pareto. Akan tetapi, sebagian besar penelitian yang telah dibahas belum menerapkan pendekatan ini. Hal tersebut dapat ditunjukkan dari penyerdehanaan yang dilakukan oleh metode-metode ini menjadi objektif tunggal melalui skalarisasi fungsi. Misalnya, untuk n fungsi objektif $f_1, f_2, \ldots, f_n$, fungsi yang dihasilkan adalah: $f = a_1 f_1 + a_2 f_2 + \cdots + a_n f_n$.

Agar dapat mengeksplorasi solusi Pareto secara efisien, algoritma MOEA (*Multi-Objective Evolutionary Algorithm*) sering digunakan dalam pendekatan ini. MOEA/D (*Multi-Objective Evolutionary Algorithm based on Decomposition*) digunakan untuk mengoptimalkan konsumsi daya server, pemborosan CPU, dan waktu propagasi (Gopu & Venkataraman, 2019), sementara NSGA-III (*Non-dominated Sorting Genetic Algorithm*) digunakan untuk meminimalkan konsumsi daya, pemborosan sumber daya, dan *network transmission delay* (Gopu dkk., 2023). Ye, Yin, dan Lin mengembangkan EEKnEA (*Energy-Efficient Knee Point-driven Evolutionary Algorithm*) untuk meminimalkan konsumsi daya, memaksimalkan *load balance*, memaksimalkan rata-rata pemanfaatan sumber daya, dan memaksimalkan rata-rata "*robustness*" server.

Tao dkk. (2016) mengembangkan BGM-BLA (*Binary Graph Matching-*

Based Bucket Code Learning Algorithm) untuk mengubah penempatan VM sehingga jumlah server aktif, komunikasi antar-VM, dan biaya migrasi VM menjadi seminimal mungkin. Sesuai dengan namanya, algoritma ini mengombinasikan algoritma *bucket-code learning* dan *binary graph matching*. BGM-BLA dibagi dalam dua tahap: pembentukan grup-grup VM dan menentukan server yang cocok sebagai tempat baru masing-masing grup tersebut. Algoritma *bucket-code learning* digunakan untuk mencari beberapa kandidat solusi optimal, sedangkan binary graph matching digunakan untuk mengevaluasi dan membandingkan kandidat-kandidat tersebut berdasarkan ketiga objektif tersebut. Kemudian, solusi tersebut dieksplorasi melalui tahap *learning* dan mutasi.

2.3 Menyelesaikan Masalah Penempatan Mesin Virtual sekaligus Masalah Penentuan Rute Jaringan

Sebagian besar metode sebelumnya juga belum memanfaatkan topologi jaringan *data center* sebagai informasi penting dalam menentukan penempatan VM, terutama untuk VM yang berkomunikasi dengan VM lain. Bahkan, metode yang mengoptimalkan metrik kinerja jaringan sering kali hanya memodelkan *data center* sebagai sekumpulan server yang dapat saling berkomunikasi dengan *bandwidth* tetap. Untuk mengatasi kekurangan ini, beberapa metode dikembangkan untuk menentukan tidak hanya penempatan VM tetapi juga rute komunikasi antar-VM. Algoritma HACOS merupakan salah satu contoh metode tersebut (Alharabe, Rakrouki & Aljohani, 2022). Akan tetapi, HACOS mengasumsikan lingkungan *cloud* yang statis. Oleh karena itu, sejumlah algoritma dirancang untuk lingkungan *cloud* yang dinamis, di mana *data center* melayani banyak *tenant* (pengguna) dengan kebutuhan sumber daya yang beragam. Setiap *tenant* dapat masuk dan keluar dari sistem pada waktu yang berbeda, sehingga algoritma-algoritma tersebut bersifat adaptif dan dijalankan secara berkala selama *data center* aktif.

Jiang dkk. (2012) menggagas sebuah algoritma heuristik yang menggunakan teknik aproksimasi rantai Markov untuk menentukan penempatan VM serta memilih *link* komunikasi yang dapat mengurangi jumlah server aktif dan rata-rata tingkat penggunaan *link*. Tidak seperti algoritma sebelumnya di mana lingkungan *cloud* bersifat statis, algoritma ini dirancang untuk lingkungan *cloud* di mana jumlah *tenant* pada waktu tertentu dimodelkan menggunakan antrean $M/M/\infty$ dan algoritma ini dijalankan setiap kali ada *tenant* yang masuk atau keluar dari sistem. Fang dkk.

(2013) mengembangkan pendekatan heuristik untuk menentukan penempatan VM dan rute komunikasi antar-VM untuk meminimalkan konsumsi daya server, biaya migrasi VM, dan *delay* komunikasi dalam jaringan. Algoritma ini dirancang khusus untuk *data center* berbasis OpenFlow dengan topologi *fat tree*. Sementara itu, Luo dkk. (2014) mengusulkan algoritma yang meminimalkan biaya komunikasi jaringan dengan memanfaatkan *minimum tree level* antar-VM berdasarkan topologi jaringan dan penempatan VM di *data center*. Algoritma ini terdiri dari dua tahap: pertama, mengevaluasi apakah total *traffic* pada setiap *switch* melebihi ambang batas yang ditentukan; kedua, memigrasikan VM ke server lain jika ambang batas terlampaui dan menentukan ulang rute komunikasi antar-VM. Algoritma ini dijalankan secara berkala pada interval waktu tertentu.

Tabel 2.1: My caption

Nama	Kegiatan	Algoritma	Perbedaan dengan peneliti
Yusuf	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmodtempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.	Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

BAB III
DASAR TEORI

3.1 *Cloud Computing*

Menurut National Institute of Standards and Technology, "*cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*." (Mell & Grance, 2011). Dalam bahasa Indonesia, *cloud computing* (komputasi awan) adalah model komputasi di mana sumber daya komputasi (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) dapat diakses melalui jaringan dari mana saja secara praktis sesuai permintaan. Sumber daya tersebut dapat dikonfigurasi, disediakan, dan dilepaskan dalam waktu singkat dengan upaya pengelolaan serta interaksi dengan penyedia layanan yang minimal.

Infrastructure as a Service (IaaS) adalah salah satu dari tiga model layanan *cloud computing*, selain *Platform as a Service* (PaaS) dan *Software as a Service* (SaaS) (Mell & Grance, 2011). IaaS menyediakan akses daya komputasi, penyimpanan, jaringan, serta sumber daya komputasi lainnya dalam bentuk abstrak atau virtual. Melalui abstraksi, pengguna dapat menyewa sumber daya tanpa perlu memiliki, mengelola, merawatnya sumber daya fisiknya secara langsung. Teknologi virtualisasi menjadi kunci terwujudnya abstraksi ini. Virtualisasi adalah proses yang mengemulasikan sumber daya komputasi fisik ke dalam bentuk abstrak atau virtual. Melalui virtualisasi, penyedia layanan dapat menawarkan akses ke berbagai sumber daya komputasi, seperti mesin virtual (*virtual machine* atau VM) yang memungkinkan pengguna menjalankan sistem operasi dan aplikasi seperti pada komputer fisik.

Kemampuan memvirtualisasi perangkat keras fisik ini dimungkinkan oleh *hypervisor*, sebuah perangkat lunak yang menjadi perantara bagi mesin virtual dalam mengakses dan berkomunikasi dengan perangkat keras milik mesin fisik (*physical machine* atau PM) tempat mesin virtual tersebut berjalan. Mesin virtual diinstal di atas *hypervisor* yang diinstal di dalam mesin fisik. *Hypervisor* berperan memetakan sumber daya fisik pada PM kepada VM yang berjalan di dalamnya sesuai dengan

kebutuhan. Selain itu, *hypervisor* juga memungkinkan PM tersebut menjalankan lebih dari satu VM dengan secara konkuren, meskipun menggunakan sistem operasi yang berbeda-beda. Meskipun demikian, *hypervisor* memastikan setiap VM terisolasi satu sama lain sehingga masing-masing dapat menjalankan program dan aplikasinya masing-masing tanpa terdampak oleh masalah dan ketidakstabilan yang terjadi pada VM lain di PM yang sama (Hill dkk., 2013). Hal ini dapat meningkatkan *fault tolerance*, kemampuan sistem untuk tetap beroperasi meskipun galat atau malfungsi.

Mesin virtual dibentuk dari sebuah *image* atau *template* yang menentukan spesifikasi seperti *virtual* CPU (vCPU), *virtual* RAM, *virtual disk*, serta *image file* dari disk tersebut. *Image* mesin virtual berupa *file* yang disimpan pada *disk*, sehingga dapat dengan mudah dibuat, dihapus, disalin, atau ditransfer ke mesin lain (Huawei Technologies Co., Ltd., 2023). Pengguna juga dapat mengambil *snapshot* VM kapan saja untuk menyimpan konfigurasi dan lingkungan (*environment*) VM yang sedang berjalan. *Snapshot* ini memungkinkan pengguna memulihkan atau menjalankan ulang lingkungan yang telah disimpan sebelumnya.

Migrasi dan replikasi VM dapat dilakukan dengan mentransfer atau menyalin *image* VM tersebut (Huawei Technologies Co., Ltd., 2023). Kemampuan migrasi VM mendukung peningkatan ketersediaan (*availability*), penghindaran bencana (*disaster avoidance*), serta pemulihan dari bencana (*disaster recovery*). Sementara itu, kemampuan replikasi VM mendukung peningkatan ketersediaan dan skalabilitas (Hill dkk., 2013). Dengan demikian, virtualisasi meningkatkan ketersediaan, penghindaran dan pemulihan dari bencana, serta skalabilitas aplikasi yang di-*deploy* oleh pengguna.

Kemampuan virtualisasi ini mendukung pendekatan *multi-tenancy*, yaitu penggunaan infrastruktur komputasi yang sama oleh beberapa pengguna sekaligus (Cloudflare, n.d.). *Multi-tenancy* dapat ditunjukkan oleh kemampuan lebih dari satu VM untuk berjalan dan mengakses sumber daya pada PM yang sama. Pendekatan *multi-tenancy* dapat mengurangi kebutuhan server pada *cloud data center* karena penyedia layanan cloud tidak perlu menyediakan satu server fisik untuk setiap pengguna (Hill dkk., 2013). Berkurangnya kebutuhan server juga mengurangi konsumsi energi listrik oleh *data center* penyedia layanan. Selain itu, *multi-tenancy* memungkinkan pemanfaatan sumber daya komputasi yang lebih efisien dibandingkan pendekatan *single-tenancy* (Cloudflare, n.d.). Dalam pendekatan *single-tenancy*, setiap pengguna disediakan akses ke server tersendiri (*dedicated server*). Pendekatan ini tidak hanya membutuhkan lebih banyak server pada *cloud data center*, tetapi juga menyebabkan sumber daya tidak digunakan secara optimal. Sumber daya yang

dialokasikan untuk satu pengguna tidak dapat diakses oleh pengguna lain, sehingga banyak sumber daya yang tidak terpakai.

3.2 Konsumsi Energi *Cloud Data Center*

Berdasarkan laporan penggunaan energi *data center* di Amerika Serikat tahun 2024 yang disusun oleh Lawrence Berkeley National Laboratory, total energi listrik yang dikonsumsi oleh server (mesin fisik) per tahun meningkat dari sekitar 30 TWh pada 2014 menjadi sekitar 100 TWh pada 2023. Sebagian besar peningkatan tersebut disebabkan oleh server AI (*Artificial Intelligence*) yang diakselerasi menggunakan GPU (*Graphical Processing Unit*), yang konsumsinya naik dari <2 TWh pada tahun 2017 menjadi >40 TWh pada 2023. Selain itu, konsumsi energi server konvensional, khususnya server *dual processor*, juga meningkat dari sekitar 30 TWh menjadi sekitar 60 TWh. Konsumsi energi server diproyeksikan akan terus meningkat hingga mencapai 240-380 TWh pada 2028.

Konsumsi listrik keseluruhan *data center* pada 2023 mencapai 176 TWh, menyumbang 4,4% dari total konsumsi energi di Amerika Serikat. Angka ini diproyeksikan meningkat menjadi sekitar 325-580 TWh pada 2028.

Pada 2023, server konvensional dan server AI secara kolektif menyumbang sekitar 60% konsumsi energi listrik di *data center*, sedangkan infrastruktur pendukung lainnya, seperti sistem pendingin dan distribusi daya, hanya menyumbang sekitar 30%. Persentase konsumsi energi oleh server diperkirakan akan meningkat menjadi sekitar 65% pada 2028 akibat semakin maraknya penggunaan AI. Dengan demikian, server tetap menjadi kontributor terbesar dalam konsumsi listrik *data center* dari 2014 hingga 2023, dan diperkirakan hingga 2028.

Pembahasan sebelumnya menunjukkan manajemen energi di *data center* menjadi isu krusial dalam komputasi awan. Konsumsi energi tidak hanya memengaruhi biaya operasional tetapi juga berdampak pada lingkungan, seperti emisi karbon yang dihasilkan oleh perangkat fisik, terutama server.

CPU merupakan komponen server dengan kontribusi konsumsi energi terbesar di antara komponen lainnya (Vasques, Moura & de Almeida, 2019). Hal ini melatarbelakangi pengembangan model-model konsumsi energi server berdasarkan tingkat penggunaan CPU (Jin dkk., 2020), salah satunya model yang dikembangkan oleh Beloglazlov, Abawajy, dan Buyya (2012). Menurut model ini, mesin fisik *idle* (menyala tetapi tidak menjalankan *task* apapun) rata-rata mengonsumsi energi

sebesar 70% dari energi yang digunakan oleh mesin fisik dengan utilisasi CPU maksimal. Selain itu, konsumsi energi server memiliki hubungan linier dengan tingkat penggunaan CPU-nya, yang dapat diperkirakan menggunakan persamaan berikut.

$$\text{PC}_j = \text{PC}_j^{\max} \cdot U_j^{\text{cpu}} + \text{PC}_j^{\text{idle}} \cdot (1 - U_j^{\text{cpu}})$$

Pada persamaan di atas,

- $\text{PC}_j \in \mathbb{R}$ merupakan konsumsi daya p_j.
- $\text{PC}_j^{\max} \in \mathbb{R}$ merupakan daya maksimum yang dikonsumsi oleh p_j ketika menggunakan CPU secara penuh.
- $\text{PC}_j^{\text{idle}} \in \mathbb{R}$ merupakan daya yang dikonsumsi oleh p_j ketika *idle*. $\text{PC}_j^{\text{idle}}$ dapat diberi nilai $0.7\text{PC}_j^{\max}$ berdasarkan penelitian yang dilakukan oleh Beloglazov, Abawajy, dan Buyya (2012).
- $U_j^{\text{cpu}} \in [0, 1]$ merupakan persentase penggunaan CPU oleh p_j.

Karena mesin fisik tetap mengonsumsi daya yang cukup besar walaupun mesin tersebut *idle*, pengurangan konsumsi daya pada *data center* dapat dilakukan dengan mematikan mesin yang *idle*. Dengan demikian, penempatan VM yang mengoptimalkan konsumsi energi akan berusaha menggunakan PM sesedikit mungkin.

3.3 Masalah Optimasi Multiobjektif

3.3.1 Definisi

Masalah optimasi multiobjektif merupakan masalah matematika yang melibatkan lebih dari satu fungsi yang disebut dengan fungsi objektif, di mana tujuan utamanya adalah mencari solusi dari himpunan tertentu yang dapat meminimalkan atau memaksimalkan beberapa fungsi objektif secara bersamaan.

Diberikan vektor fungsi objektif $\mathbf{f} = [f_1, f_2, \cdots, f_m]$ yang terdiri dari m fungsi objektif, yaitu $f_1, f_2, \cdots, f_m$, dengan $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Untuk setiap $i = 1, 2, \cdots, m$, fungsi objektif f_i memetakan vektor keputusan atau solusi $\mathbf{x}$ ke nilai objektif dalam $\mathbb{R}$.

Secara umum, masalah optimasi multiobjektif dapat dinyatakan sebagai berikut :

$$\max_{\mathbf{x} \in S} \mathbf{f}(\mathbf{x})$$

Di sini, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]$ merupakan vektor nilai objektif dari $\mathbf{x}$, di mana elemen ke-i menyatakan nilai fungsi objektif $f_i(\mathbf{x})$, untuk setiap $i = 1, 2, \ldots, m$. Jika hanya terdapat satu fungsi objektif ($m = 1$), maka masalah ini menjadi masalah optimasi berobjektif tunggal. Sebaliknya, jika $m \geq 2$, masalah ini disebut sebagai masalah optimasi multiobjektif.

Perlu dicatat bahwa tidak semua masalah optimasi multiobjektif melibatkan maksimalisasi seluruh fungsi dalam vektor objektif. Dalam beberapa kasus, masalah ini dapat dirumuskan sebagai pencarian solusi yang meminimalkan nilai satu atau lebih fungsi objektif sementara nilai fungsi objektif yang lain dimaksimalkan. Agar selaras dengan bentuk umum di atas, setiap fungsi objektif f_i yang hendak diminimalkan dapat diubah menjadi fungsi yang dimaksimalkan dengan transformasi $f_i \rightarrow af_i + b$, di mana $a < 0$. Dengan demikian, tanpa menghilangkan keumuman, setiap masalah optimasi multiobjektif dapat dinyatakan sebagai masalah maksimalisasi.

Vektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ disebut solusi atau vektor keputusan dengan elemen ke-i berupa variabel keputusan x_i, untuk setiap $i = 1, 2, \ldots, n$. Himpunan $S \subseteq \mathbb{R}^n$ disebut sebagai ruang keputusan, yang terdiri dari semua solusi feasibel yang memenuhi batasan berupa pertidaksamaan atau persamaan tertentu.

Secara matematis,

$$S = \{\mathbf{x} \in \mathbb{R}^n \mid \forall j = 1, 2, \ldots, p,\ \forall k = 1, 2, \ldots, q,\ g_j(\mathbf{x}) \geq 0 \text{ dan } h_k(\mathbf{x}) = 0\}$$

dengan $p, q \in \mathbb{N}$. Himpunan $\mathbf{f}(S) \subset \mathbb{R}^M$ disebut sebagai ruang objektif, yang merupakan himpunan dari semua vektor nilai objektif yang diperoleh dari solusi feasibel dalam S.

Masalah optimasi multiobjektif di atas dapat ditulis dalam bentuk berikut :

$$\begin{aligned}
\text{Minimalkan} \quad & \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})] && (3.1)\\
\text{dengan} \quad & g_j(\mathbf{x}) \geq 0, \quad j = 1, 2, \ldots, p && (3.2)\\
& h_k(\mathbf{x}) = 0, \quad k = 1, 2, \ldots, q && (3.3)\\
& \mathbf{x} \in \mathbb{R}^n && (3.4)
\end{aligned}$$

Berdasarkan kardinalitas domainnya, variabel keputusan dapat dibagi menjadi dua jenis: diskret dan kontinu. Variabel keputusan diskret memiliki domain yang

merupakan himpunan terhitung (*countable set*), seperti $\{0, 1\}$ (biasa disebut sebagai variabel biner), $\mathbb{N}$, atau $\mathbb{Z}$. Sedangkan variabel keputusan kontinu memiliki domain yang merupakan himpunan takterhitung (*uncountable set*), seperti $\mathbb{R}$, $[0, \infty)$, atau $[0, 1]$.

3.3.2 Pemrograman Linier

Pemrograman linier atau LP (*linear programming*) adalah sebuah kelas masalah optimasi di mana semua fungsi objektif dan kendalanya bersifat linier. Oleh karena itu, LP dapat dipandang sebagai bentuk khusus dari masalah optimasi multiobjektif. LP juga memiliki beberapa bentuk khusus berdasarkan jenis variabel keputusannya. Salah satunya adalah MILP (*Mixed Integer Linear Programming*) yang menggunakan variabel keputusan diskret sekaligus kontinu.

3.3.3 Optimalitas Pareto

Pada masalah minimalisasi berobjektif tunggal, solusi yang paling optimal adalah solusi dengan nilai objektif paling rendah. Hal ini serupa dengan maksimalisasi berobjektif tunggal. Karena fungsi objektif memiliki nilai dalam himpunan bilangan riil, setiap nilai objektif dapat diurutkan secara linier. Selain itu, nilai maksimum dan minimum yang dapat dicapai oleh fungsi objektif tidak lebih dari satu.

Akan tetapi, membandingkan keoptimalan dua buah solusi pada masalah multiobjektif tidak semudah itu, terlebih lagi dalam menentukan solusi paling optimal. Dalam beberapa kasus, sering kali dijumpai sepasang solusi, $\mathbf{x}^{(1)}$ dan $\mathbf{x}^{(2)}$, di mana solusi $\mathbf{x}^{(1)}$ lebih optimal daripada solusi $\mathbf{x}^{(2)}$ menurut satu atau lebih fungsi objektif, sementara solusi $\mathbf{x}^{(2)}$ lebih optimal daripada solusi $\mathbf{x}^{(1)}$ menurut fungsi objektif lainnya.

Sebagai contoh, misalkan terdapat sepasang solusi, $\mathbf{x}^{(1)}$ dan $\mathbf{x}^{(2)}$, di mana $\mathbf{f}(\mathbf{x}^{(1)}) = (1, 4)^T$ dan $\mathbf{f}(\mathbf{x}^{(2)}) = (2, 3)^T$. Solusi $\mathbf{x}^{(1)}$ tidak bisa dibandingkan terhadap solusi $\mathbf{x}^{(2)}$ karena menurut f_1, solusi $\mathbf{x}^{(1)}$ lebih optimal ($1 < 2$), tetapi menurut f_2, solusi $\mathbf{x}^{(2)}$ lah yang lebih optimal ($4 > 3$).

Membandingkan nilai dua buah vektor elemen demi elemen, seperti pada contoh sebelumnya, tidak menghasilkan pengurutan total (*total ordering*), melainkan pengurutan parsial (*partial ordering*) (kecuali vektor berdimensi nol dan satu). Oleh karena itu, mencari vektor nilai objektif "maksimum" atau "minimum" belum tentu

dapat dilakukan.

Selain itu, dalam beberapa masalah multiobjektif, pengoptimalan setiap fungsi objektif tidak selalu dapat dilakukan secara bersamaan. Sering kali ditemukan solusi yang hanya mengoptimalkan sebagian fungsi objektif, tetapi ketika solusi di sekitarnya berusaha mengoptimalkan fungsi yang sebelumnya kurang optimal, nilai fungsi yang semula optimal justru menjadi kurang optimal.

Salah satu strategi klasik untuk memecahkan masalah optimasi multiobjektif adalah mereduksinya menjadi masalah baru dengan objektif tunggal, sehingga dapat diselesaikan dengan metode yang lebih sederhana. Baik masalah asli maupun masalah baru memiliki komponen yang sama, kecuali fungsi objektifnya. Pada masalah baru ini, fungsi objektif dibentuk dengan menggabungkan semua fungsi objektif dari masalah asli menjadi satu fungsi.

Terdapat tiga pendekatan dalam pembentukan fungsi objektif baru ini: pendekatan penjumlahan berbobot, pendekatan metrik L_p atau fungsi jarak (misalnya jarak Manhattan, jarak Euclid, atau jarak Chebyshev), dan pendekatan *boundary intersection*. Meskipun reduksi ini mampu memperoleh satu solusi optimal Pareto, metode ini masih memiliki kelemahan utama, yakni sensitivitas terhadap bobot yang diberikan kepada tiap fungsi objektif. Pengambil keputusan (*decision maker*) harus menetapkan bobot yang sesuai dengan karakteristik masalah untuk memperoleh solusi optimal. Namun, pemberian bobot juga memerlukan pemahaman tentang skala prioritas tiap fungsi objektif, karena semakin besar bobot yang diberikan kepada suatu fungsi objektif, semakin tinggi prioritas fungsi tersebut sehingga nilai fungsi objektif tersebut mungkin menjadi lebih sensitif terhadap perubahan solusi.

Untuk mengatasi kendala tersebut, sebarang dua solusi dibandingkan berdasarkan optimalitas Pareto. Diberikan dua vektor nilai objektif $\mathbf{z}^{(1)}, \mathbf{z}^{(2)} \in \mathbb{R}^m$, di mana $\mathbf{z}^{(1)} = (z_1^{(1)}, z_2^{(1)}, \ldots, z_m^{(1)})$ dan $\mathbf{z}^{(2)} = (z_1^{(2)}, z_2^{(2)}, \ldots, z_m^{(2)})$, vektor $\mathbf{z}^{(1)}$ disebut mendominasi vektor $\mathbf{z}^{(2)}$ (ditulis $\mathbf{z}^{(1)} \succ \mathbf{z}^{(2)}$) jika berlaku dua kondisi berikut:

1. $z_i^{(1)} \geq z_i^{(2)}$ untuk setiap $i = 1, 2, \ldots, m$, dan
2. Terdapat paling tidak satu indeks $j = 1, 2, \ldots, m$ dengan $z_j^{(1)} > z_j^{(2)}$.

Dengan kata lain, $\mathbf{z}^{(1)}$ mendominasi $\mathbf{z}^{(2)}$ apabila untuk setiap fungsi objektif, nilainya pada $\mathbf{z}^{(1)}$ tidak lebih buruk daripada $\mathbf{z}^{(2)}$, dan setidaknya terdapat satu fungsi objektif di mana $\mathbf{z}^{(1)}$ lebih unggul.

Berdasarkan dominasi antar vektor nilai objektif, solusi $\mathbf{x}^{(1)}$ disebut mendominasi solusi $\mathbf{x}^{(2)}$ berdasarkan vektor objektif $\mathbf{f}$ (ditulis $\mathbf{x}^{(1)} \succ_{\mathbf{f}} \mathbf{x}^{(2)}$) jika

$\mathbf{f}(\mathbf{x}^{(1)}) \succ \mathbf{f}(\mathbf{x}^{(2)})$. Sebuah solusi $\mathbf{x}^*$ dikatakan optimal Pareto (*Pareto optimal*) terhadap himpunan S jika tidak ada solusi lain dalam S yang mendominasi $\mathbf{x}^*$. Vektor $\mathbf{f}(\mathbf{x}^*)$ disebut sebagai vektor objektif optimal Pareto (*Pareto optimal objective vector*).

Keistimewaan solusi optimal Pareto adalah bahwa upaya mengoptimalkan satu fungsi objektif akan menyebabkan penurunan performa pada setidaknya satu fungsi objektif lainnya. Himpunan semua solusi optimal Pareto terhadap S disebut himpunan Pareto (*Pareto set*) pada S , sedangkan himpunan semua vektor objektif optimal Pareto-nya disebut sebagai *Pareto front* pada S.

Perlu diperhatikan bahwa optimalitas Pareto tidak menghasilkan pengurutan total dan tidak menghilangkan konflik antar fungsi objektif. Meskipun tidak menghasilkan pengurutan total, konsep ini tetap berguna karena dalam banyak masalah multiobjektif, tidak ada solusi tunggal yang optimal secara mutlak di setiap fungsi objektif. Selain itu, konsep ini mengakomodasi keberadaan konflik tersebut dengan menentukan himpunan solusi yang tidak bisa diperbaiki lebih lanjut tanpa mengorbankan aspek yang lain. Metode penentuan ini lebih sistematis daripada hanya membandingkan elemen demi demi tanpa aturan dominasi. Oleh karena itu, *Pareto front* bukanlah solusi akhir, melainkan kumpulan solusi yang harus dianalisis lebih lanjut berdasarkan preferensi tertentu.

3.4 Penempatan Mesin Virtual pada Mesin Fisik dalam *Data Center*

3.4.1 Asumsi

Sebuah *cloud data center* memiliki sejumlah PM (*physical machine* atau mesin fisik) dengan kapasitas sumber daya yang berbeda, terutama CPU dan memori. Sejumlah VM (*virtual machine* atau mesin virtual) dengan kebutuhan sumber daya yang bervariasi dipesan oleh pengguna layanan *cloud*. Setiap VM hanya dapat ditempatkan pada satu PM, sementara satu PM dapat menjalankan lebih dari satu VM. Setiap VM ditempatkan dengan memperhatikan kapasitas sumber daya yang tersisa di setiap PM.

Karena konsumsi energi sebuah PM berbanding lurus dengan penggunaan CPU-nya (Beloglazov, Abawajy Buyya, 2012), penempatan VM harus dioptimalkan supaya total konsumsi energi seluruh PM dalam *data center* menjadi seminimal mungkin.

Sisa sumber daya yang tersedia pada setiap PM dapat sangat bervariasi

tergantung pada solusi penempatan VM yang dipilih. Untuk memaksimalkan pemanfaatan sumber daya, metrik yang dikembangkan oleh Gao, dkk. (2013) digunakan untuk menghitung potensi biaya pemborosan sumber daya. Penempatan VM kemudian ditentukan untuk meminimalkan pemborosan berdasarkan metrik ini.

Dengan demikian, minimalisasi konsumsi energi serta pemborosan sumber daya menjadi objektif utama dalam model optimasi penempatan VM pada penelitian ini.

Masalah penempatan VM dapat dipandang sebagai perluasan masalah *bin packing*. Dalam masalah *bin packing* klasik, sejumlah objek dengan ukuran tertentu harus ditempatkan ke dalam wadah (*bin*) berkapasitas terbatas, dengan tujuan meminimalkan jumlah wadah yang digunakan. Pada *bin packing* standar, semua wadah memiliki kapasitas yang sama dan setiap objek hanya memiliki satu ukuran. Namun, dalam konteks penempatan VM, setiap VM dan PM memiliki dua dimensi utama: CPU dan memori. Selain itu, setiap PM memiliki kapasitas yang bervariasi. Oleh karena itu, masalah penempatan VM merupakan salah bentuk *Vector Bin Packing Problem* (VBPP).

Masalah *bin packing* diketahui memiliki kompleksitas *NP-complete*. Karena masalah penempatan VM dimodelkan sebagai VBPP, masalah penempatan VM juga termasuk dalam kelas masalah *NP-complete* (Fatima dkk., 2018).

3.4.2 Notasi

Misalkan *data center* memiliki N_P buah PM dan terdapat N_V buah VM yang harus ditempatkan. Definisikan $P = \{p_1, p_2, \ldots, p_n\}$ dan $V = \{v_1, v_2, \ldots, v_m\}$ masing-masing sebagai himpunan PM dan himpunan VM pada *data center*, di mana $n = N_P$ dan $m = N_V$.

Penempatan VM pada PM dinyatakan sebagai pemetaan $\pi : V \to P$, di mana $\pi(v) = p$ jika dan hanya jika VM $v \in V$ ditempatkan pada PM $p \in P$.

Kebutuhan CPU dan memori dari VM $v_i \in V$ masing-masing dinotasikan sebagai v_i^{cpu} dan v_i^{mem}, sedangkan kapasitas CPU dan memori yang tersedia pada PM $p_j \in P$ dinotasikan sebagai p_j^{cpu} dan p_j^{mem}

Rasio utilisasi CPU dan memori oleh PM $p_j \in P$ didefinisikan sebagai:

$$U_j^{\text{cpu}} := \frac{\sum_{v_i \in V \,:\, \pi(v_i) = p_j} v_i^{\text{cpu}}}{p_j^{\text{cpu}}}$$

$$U_j^{\text{mem}} := \frac{\sum_{v_i \in V \,:\, \pi(v_i)=p_j} v_i^{\text{mem}}}{p_j^{\text{mem}}}$$

, sedangkan rasio sisa CPU dan memori pada PM $p_j \in P$ didefinisikan sebagai $L_j^{\text{cpu}} := 1 - U_j^{\text{cpu}}$ dan $L_j^{\text{mem}} := 1 - U_j^{\text{mem}}$

PM $p_j \in P$ mengonsumsi energi sebesar $\text{PC}_j^{\text{idle}}$ ketika beroperasi tanpa menjalankan VM apapun dan mengonsumsi energi sebesar PC_j^{max} ketika p_j ketika CPU digunakan secara penuh. Berdasarkan metrik konsumsi energi PM yang dikembangkan oleh Beloglazov, Abawajy, dan Buyya (2012), konsumsi energi PM p_j ketika CPU digunakan dengan rasio utilisasi U_j^{cpu} didefinsikan sebagai:

$$\text{PC}_j := \text{PC}_j^{\max} \cdot U_j^{\text{cpu}} + \text{PC}_j^{\text{idle}} \cdot (1 - U_j^{\text{cpu}})$$

Total konsumsi energi oleh seluruh PM pada *data center* dapat dihitung sebagai berikut:

$$\text{PC}_{\text{sum}} = \sum_{j=1}^{N_P} \text{PC}_j y_j$$

di mana

$$y_j = \begin{cases} 1 & \text{, jika terdapat VM } v \in V \text{ di mana } \pi(v) = p_j \\ 0 & \text{, jika sebaliknya} \end{cases}$$

Pemborosan sumber daya oleh PM $p_j \in P$ menurut Gao dkk. (2013) didefinsikan sebagai:

$$\text{RW}_j := \frac{|L_j^{\text{cpu}} - L_j^{\text{mem}}| + \epsilon}{U_j^{\text{cpu}} + U_j^{\text{mem}}}$$

, di mana $\epsilon > 0$ merupakan bilangan positif yang sangat kecil. Total pemborosan sumber daya pada *data center* dapat dihitung sebagai berikut:

$$\text{RW}_{\text{sum}} = \sum_{j=1}^{N_P} \text{RW}_j y_j$$

3.4.3 Formulasi Masalah

Berdasarkan deskripsi masalah di atas, masalah penempatan VM dapat dinyatakan secara matematis secara berikut:

Tabel 3.1:

Minimalkan	PC_{sum}	(O1)	
Minimalkan	RW_{sum}	(O2)	
dengan syarat	$\sum_{j=1}^{N_P} x_{ij} = 1$	(V1)	$i = 1, 2, \ldots, N_V$
	$\sum_{i=1}^{N_V} v_i^{\text{cpu}} x_{ij} \leq p_i^{\text{cpu}}$	(V2)	$j = 1, 2, \ldots, N_P$
	$\sum_{i=1}^{N_V} v_i^{\text{mem}} x_{ij} \leq p_i^{\text{mem}}$	(V3)	$j = 1, 2, \ldots, N_P$
	$y_j = \begin{cases} 0 & \text{, jika } \sum_{i=1}^{N_V} x_{ij} = 0 \\ 1 & \text{, jika } \sum_{i=1}^{N_V} x_{ij} > 0 \end{cases}$	(V4)	$j = 1, 2, \ldots, N_P$
	$x_{ij} \in \{0, 1\}$	(V5)	$i = 1, 2, \ldots, N_V$

Model di atas mempertimbangkan dua objektif, yaitu meminimalkan total konsumsi energi seluruh PM (O1), serta meminimalkan total pemborosan sumber daya oleh seluruh PM (O2). Kedua objektif didefinisikan menggunakan variabel y_j pada batasan (V5) untuk memastikan bahwa total dihitung hanya menggunakan PM yang aktif.

Batasan (V1) memastikan bahwa setiap VM ditempatkan pada satu PM saja. Batasan (V2) dan (V3) memastikan bahwa total CPU dan memori yang digunakan oleh semua VM yang ditempatkan pada suatu PM tidak melebihi kapasitas CPU dan memori yang dimiliki PM tersebut. Ketiga batasan tersebut menggunakan variabel indikator x_{ij} pada batasan (V5) yang bernilai 1 jika $\pi(v_i) = p_j$ dan bernilai 0 jika sebaliknya . Terakhir, batasan (V4) memastikan bahwa PM diaktifkan jika dan hanya terdapat paling tidak satu VM yang dipetakan kepada PM tersebut.

Model di atas menggunakan x_{ij} sebagai variabel keputusan, untuk setiap $i = 1, \ldots, N_V$ dan $j = 1, \ldots, N_P$. Dengan demikian, jumlah variabel keputusan dalam model ini adalah $N_P \cdot N_V$

3.5 Penentuan Rute dalam Jaringan *Data Center*

3.5.1 Asumsi

Cloud data center memiliki jaringan yang menghubungkan PM dan perangkat jaringan seperti *switch* dan *router* melalui beberapa *link*. Mesin-mesin virtual yang sudah ditempatkan pada PM-nya masing-masing akan saling berkomunikasi dengan kebutuhan *bandwidth* tertentu sebelum proses pemetaan dilakukan. Jika dua VM ditempatkan dalam PM yang sama, komunikasi mereka dapat dilakukan di dalam PM tanpa melalui jaringan. Namun, apabila dua buah VM ditempatkan pada PM yang berbeda, keduanya harus berkomunikasi melewati jaringan dengan membentuk koneksi antara PM tempat mereka dipetakan. Akibatnya, jumlah koneksi serta alokasi *bandwidth* yang diperlukan bergantung pada hasil pemetaan VM.

Subbab ini membahas notasi serta model optimasi yang digunakan dalam penentuan rute komunikasi dalam jaringan. Dalam model ini, lalu lintas antara setiap pasang PM yang berkomunikasi akan dialokasikan paling banyak ke k buah rute, di mana nilai k ditentukan oleh pengambil keputusan. Dengan demikian, masalah optimasi ini dapat dipandang sebagai masalah *k-splittable multicommodity flow*. Masalah ini dikenal memiliki kompleksitas *NP-complete*.

Dalam praktiknya, mengalokasikan *bandwidth* secara penuh untuk setiap koneksi tidak dapat dilakukan tidak selalu memungkinkan, terlepas dari bagaimana VM ditempatkan dalam PM. Oleh karena itu, salah satu objektif dalam model optimasi ini adalah memaksimalkan total bandwidth yang dapat dialokasikan dalam jaringan. Dalam konteks masalah multicommodity flow, objektif ini berkaitan dengan masalah *maximum flow*. Selain itu, meminimalkan *delay* komunikasi antar PM juga menjadi objektif penting model ini untuk meningkatkan efisiensi transmisi data. Dalam konteks masalah multicommodity flow, objektif ini berkaitan dengan masalah *minimum cost*.

Dalam infrastruktur *cloud data center* modern, teknologi *Software Defined Networking* (SDN) memungkinkan pengelolaan jaringan yang lebih fleksibel dan terpusat. SDN memisahkan *control plane* dari *data plane*, sehingga kontrol lalu lintas jaringan dapat dilakukan melalui *controller* yang memiliki visibilitas penuh terhadap topologi jaringan. Dengan pendekatan ini, rute komunikasi antar *Physical Machine* (PM) dapat ditentukan secara eksplisit dan dikonfigurasi sesuai kebutuhan.

Selain itu, teknologi *Multipath TCP* (MP-TCP) memungkinkan pemisahan lalu lintas komunikasi ke dalam beberapa rute secara bersamaan (Ford dkk., 2011).

Beberapa jaringan yang berhasil mengimplementasikan *multicommodity flow* sebagai solusi *traffic engineering* dalam menggunakan SDN adalah B4, jaringan WAN pribadi milik Google (Hong dkk., 2018), dan SWAN, jaringan WAN pribadi milik Microsoft (Hong dkk., 2013).

Oleh karena itu, asumsi dalam model yang diajukanyakni bahwa rute komunikasi dapat ditentukan dan lalu lintas komunikasi dapat dibagi menjadi lebih dari satu rute sejalan dengan praktik yang diterapkan dalam *cloud data center* berbasis SDN.

3.5.2 Notasi

Misalkan $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ adalah graf berarah tanpa *loop* yang mewakili topologi jaringan *data center*, di mana $\mathcal{N}$ dan $\mathcal{L}$ masing-masing merupakan himpunan *node* dan *link*. $\mathcal{N}$ mencakup semua mesin fisik dalam himpunan P beserta perangkat jaringan. Setiap link $l = (u, v) \in \mathcal{L}$, menghubungkan dua *node* $u, v \in \mathcal{N}$, dengan u sebagai *node* asal dan v sebagai *node* tujuan. Kapasitas dan *delay link* $l \in \mathcal{L}$ masing-masing diberi notasi $c_l \in \mathbb{N}$ dan $\delta_l \in \mathbb{N}$.

Misalkan $b_{\text{VM}}(v_i, v_j) \in \mathbb{N}$ adalah besar *bandwidth* yang diminta untuk komunikasi antara VM $v_i \in V$ dan $v_j \in V$ dan $b_{\text{PM}}(p_i, p_j) \in \mathbb{N}$ adalah besar *bandwidth* yang dialokasikan untuk komunikasi antara PM $p_i \in P$ dan $p_j \in P$. Maka, $b_{\text{PM}}(p_i, p_j)$ didefinisikan sebagai

$$b_{PM}(p_i, p_j) = \begin{cases} 0 & \text{, jika } p_i = p_j \\ \displaystyle\sum_{\substack{(v_i, v_j) \in V \times V \\ \pi(v_i) = p_i \,\wedge\, \pi(v_j) = p_j}} b_{\text{VM}}(v_i, v_j) & \text{, jika } p_i \neq p_j \end{cases}$$

Rute dari *node* $u \in \mathcal{N}$ ke *node* $v \in \mathcal{N}$ didefinisikan sebagai barisan *link*

$$r = (l_1, l_2, \ldots, l_n)$$

, di mana

- $l_i = (u_{i-1}, u_i) \in \mathcal{L}$, untuk $i = 1, 2, \ldots, n$,
- $u_0 = u$ adalah *node* asal, sedangkan $u_n = v$ adalah *node* tujuan,
- tidak ada *node* yang berulang dalam rute, yaitu $u_i \neq u_j$ untuk setiap $i \neq j$

- panjang rute adalah jumlah *link*-nya, yaitu n

Misalkan $\mathcal{R}^*(p', p'')$ adalah himpunan semua rute dari PM $p' \in P$ ke PM $p'' \in P$ yang ada dalam jaringan $\mathcal{G}$, dengan $k' = |\mathcal{R}^*(p', p'')|$. Karena komunikasi antara p' dan p'' dilakukan melalui paling banyak k buah rute, maka hanya $k'' \leq k$ rute yang benar-benar digunakan.

Untuk setiap rute $r \in \mathcal{R}^*(p', p'')$ misalkan b_r adalah besar *bandwidth* yang dialokasikan pada rute tersebut, dengan ketentuan:

$$b_r = \begin{cases} \text{nilai positif}, & \text{jika } r \text{ termasuk dalam rute yang digunakan} \\ 0, & \text{jika } r \text{ tidak digunakan.} \end{cases}$$

Oleh karena itu, total *bandwidth* yang dialokasikan untuk komunikasi antara p' dan p'' adalah:

$$b_{\text{PM}}(p', p'') = \sum_{r \in \mathcal{R}^*(p', p'')} b_r$$

Definsikan akumulasi *delay* rute r sebagai

$$\Delta_r = \sum_{l \in \text{range}(r)} \delta_l$$

, di mana range(r) adalah himpunan semua *link* yang digunakan pada rute r. Terdapat dua objektif yang hendak dioptimalkan dalam masalah ini, yaitu memaksimalkan total alokasi *bandwidth* dalam jaringan dan meminimalkan rata-rata *delay* antara semua rute yang terpakai. Total alokasi *bandwidth* dapat didefinisikan sebagai:

$$\text{BW}_{\text{sum}} := \sum_{(p_i, p_j) \in P \times P} \sum_{r \in \mathcal{R}^*(p_i, p_j)} b_r$$

, sedangkan rata-rata *delay* dapat didefinisikan sebagai:

$$\text{D}_{\text{mean}} := \frac{\displaystyle\sum_{(p_i, p_j) \in P \times P} \sum_{r \in \mathcal{R}^*(p_i, p_j)} \Delta_r \cdot b_r}{\displaystyle\sum_{(p_i, p_j) \in P \times P} \sum_{r \in \mathcal{R}^*(p_i, p_j)} [b_r > 0]}$$

$\Delta_r \cdot b_r(p_i, p_j)$ menyatakan delay yang dialami untuk mentransmisikan data

sebesar $b_r(p_i, p_j)$ dari PM p_i ke PM p_j melalui rute r. Dengan demikian, pembilang pada definisi D_{mean} di atas menyatakan total *delay* yang dialami oleh setiap komunikasi.

Notasi $[P]$ adalah notasi kurung Iverson yang melambangkan nilai kebenaran pernyataan P, di mana $[P] = 1$ apabila P benar dan $[P] = 0$ apabila P salah. Dengan demikian, penyebut pada definisi D_{mean} menyatakan jumlah rute yang dipakai.

3.5.3 Formulasi Masalah

Secara matematis, masalah penentuan rute dapat dinyatakan sebagai masalah optimasi multiobjektif berikut:

Tabel 3.2:

Maksimalkan	BW_{sum}	(O3)
Minimalkan	D_{mean}	(O4)
dengan syarat	$b_r \geq \dfrac{x_r}{M}$	(R1)
	$b_r \leq x_r \cdot b_{\text{PM}}(p_i, p_j)$	(R2)
	$\displaystyle\sum_{r \in \mathcal{R}^*(p_i, p_j)} x_r \leq k$	(R3)
	$\displaystyle\sum_{(p_i, p_j) \in P \times P} \; \sum_{r \in \mathcal{R}^*(p_i, p_j) \,:\, l \in \text{range}(r)} b_r \leq c_l$	(R4)
	$b_r \in \mathbb{R}$	(R5)
	$x_r \in \{0, 1\}$	(R6)

Model di atas mempertimbangkan dua objektif, yaitu memaksimalkan alokasi *bandwidth* (O3), serta meminimalkan *delay* rata-rata di antara semua komunikasi (O4).

Batasan (R1) memastikan bahwa setiap PM tidak mengirimkan data dalam jumlah negatif. Batasan (R2) membatasi data yang dikirim antara pasangan PM supaya tidak melebihi permintaan alokasi *bandwidth*. Batasan (R3) memastikan bahwa data hanya dapat ditransmisikan melalui maksimal k rute. Batasan ini menggunakan variabel indikator x_r, yang bernilai 1 jika rute r digunakan untuk berkomunikasi dan 0 jika tidak, sebagaimana tercantum dalam batasan (R6). Batasan (R4) memastikan bahwa total *bandwidth* yang digunakan untuk transmisi data melalui suatu *link* tidak melebihi kapasitasnya. Terakhir, batasan (R6) menjamin bahwa

bandwidth yang dialokasikan untuk setiap rute bernilai bilangan riil.

Model di atas menggunakan $b_r(p_i, p_j)$ dan x_r sebagai variabel keputusan, untuk setiap $p_i, p_j \in P$ dan $r \in \mathcal{R}^*(p_i, p_j)$. Dengan demikian, jumlah variabel keputusan dalam model ini adalah $O(|P|^2 R)$, dengan R merupakan total rute yang ada di dalam jaringan.

Namun, karena tidak semua pasangan PM saling berkomunikasi, semua variabel $b_r(p_i, p_j)$ dapat diberi nilai 0 untuk setiap rute r dari p_i ke p_j jika p_i tidak berkomunikasi dengan p_j. Hal in dapat mengurangi jumlah variabel keputusan secara signifikan.

Selain itu, dalam implementasi pencarian solusi menggunakan algoritma genetika, hanya k rute dengan akumulasi *delay* terkecil yang dipertimbangkan sebagai variabel keputusan. Rute-rute ini ditentukan sebelum algoritma berjalan menggunakan strategi khusus yang sesuai topologi jaringan yang hendak dioptimalkan. Apalagi tidak memiliki strategi khusus, pengambil keputusan dapat menggunakan strategi yang lebih umum. Sebagai contoh, pengambil keputusan dapat memilih k rute dengan *delay* terkecil menggunakan algoritma *k-shortest path* (KSP), seperti algoritma Yen.

3.6 Algoritma Genetika

3.7 *Nondominated Sorting Genetic Algorithm III* (NSGA-III)

Nondominated Sorted Genetic Algorithm III (NSGA-III) merupakan versi ketiga NSGA yang dikembangkan oleh Deb dan Jain (2014). Ketiga versi algoritma ini memiliki kerangka kerja yang serupa, terutama dalam penggunaan teknik *non-dominated sorting* (pengurutan takterdominasi) untuk mengevaluasi populasi yang dihasilkan. *Non-dominated sorting* adalah metode pemeringkatan individu sebagai solusi atau vektor objektif dalam optimasi multiobjektif berdasarkan prinsip dominasi Pareto. Seperti dijelaskan pada subbab sebelumnya, individu $\mathbf{x}^{(1)}$ dikatakan mendominasi individu $\mathbf{x}^{(2)}$ jika dan hanya jika tidak ada satu pun objektif pada $\mathbf{x}^{(1)}$ yang lebih buruk daripada objektif yang sama pada $\mathbf{x}^{(2)}$, serta terdapat setidaknya satu objektif pada $\mathbf{x}^{(1)}$ yang lebih baik dibandingkan objektif yang sama pada $\mathbf{x}^{(2)}$. Dalam hal ini, individu $\mathbf{x}^{(1)}$ memiliki peringkat lebih tinggi daripada individu $\mathbf{x}^{(2)}$. Individu-individu dengan peringkat yang sama dikelompokkan ke dalam *front* takterdominasi yang sama. Selain itu, individu dalam *front* takterdominasi yang sama memiliki

setidaknya dua objektif yang saling berkonflik, sehingga tidak ada individu yang mendominasi solusi lainnya.

Misalkan *front* takterdominasi peringkat i dalam populasi P dilambangkan sebagai $F_i \subseteq P$. *Front* pertama, F_1 berisi semua solusi yang tidak didominasi oleh solusi mana pun di P. Selanjutnya, *front* F_2 berisi semua solusi takterdominasi pada $P \setminus F_1$. Demikian pula dengan *front* F_3 yang terdiri dari semua solusi takterdominasi pada $P \setminus (F_1 \cup F_2)$, dan seterusnya. Secara umum, untuk $i > 1$, F_i adalah himpunan solusi takterdominasi dalam $P \setminus (\bigcup_{j=1}^{i-1} F_j)$.

Sebagai contoh, misalkan $P = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}, \mathbf{x}^{(5)}, \mathbf{x}^{(6)}\}$ adalah populasi dengan enam individu untuk suatu masalah optimasi dengan tiga objektif, di mana

$$\mathbf{z}^{(1)} = \mathbf{f}(\mathbf{x}^{(1)}) = \begin{bmatrix} 0.67228 & 0.29762 & 0.37744 \end{bmatrix}$$
$$\mathbf{z}^{(2)} = \mathbf{f}(\mathbf{x}^{(2)}) = \begin{bmatrix} 0.48808 & 0.04670 & 0.49415 \end{bmatrix}$$
$$\mathbf{z}^{(3)} = \mathbf{f}(\mathbf{x}^{(3)}) = \begin{bmatrix} 0.82550 & 0.99063 & 0.92895 \end{bmatrix}$$
$$\mathbf{z}^{(4)} = \mathbf{f}(\mathbf{x}^{(4)}) = \begin{bmatrix} 0.03145 & 0.00683 & 0.39545 \end{bmatrix}$$
$$\mathbf{z}^{(5)} = \mathbf{f}(\mathbf{x}^{(5)}) = \begin{bmatrix} 0.80805 & 0.76979 & 0.97396 \end{bmatrix}$$
$$\mathbf{z}^{(6)} = \mathbf{f}(\mathbf{x}^{(6)}) = \begin{bmatrix} 0.56562 & 0.74677 & 0.52441 \end{bmatrix}$$

Berikut adalah relasi dominasi antar individu di P: 1. $\mathbf{z}^{(1)} \prec \mathbf{z}^{(3)}$ 2. $\mathbf{z}^{(1)} \prec \mathbf{z}^{(5)}$ 3. $\mathbf{z}^{(2)} \prec \mathbf{z}^{(3)}$ 4. $\mathbf{z}^{(2)} \prec \mathbf{z}^{(5)}$ 5. $\mathbf{z}^{(2)} \prec \mathbf{z}^{(6)}$ 6. $\mathbf{z}^{(4)} \prec \mathbf{z}^{(2)}$ 7. $\mathbf{z}^{(4)} \prec \mathbf{z}^{(3)}$ 8. $\mathbf{z}^{(4)} \prec \mathbf{z}^{(5)}$ 9. $\mathbf{z}^{(4)} \prec \mathbf{z}^{(6)}$ 10. $\mathbf{z}^{(6)} \prec \mathbf{z}^{(3)}$ 11. $\mathbf{z}^{(6)} \prec \mathbf{z}^{(5)}$ Perhatikan bahwa $\mathbf{x}^{(1)}$ dan $\mathbf{x}^{(4)}$ tidak didominasi oleh individu manapun di P. Oleh karena itu, $\mathbf{x}^{(1)}$ dan $\mathbf{x}^{(4)}$ termasuk ke dalam front takterdominasi peringkat pertama. Selain itu, $\mathbf{x}^{(3)}$ dan $\mathbf{x}^{(5)}$ tidak mendominasi individu manapun, sehingga $\mathbf{x}^{(3)}$ dan $\mathbf{x}^{(5)}$ termasuk ke dalam front takterdominasi peringkat terakhir. Dengan demikian, 1. $F_1 = \{\mathbf{x}^{(1)}, \mathbf{x}^{(4)}\}$ 2. $F_2 = \{\mathbf{x}^{(2)}\}$ 3. $F_3 = \{\mathbf{x}^{(6)}\}$ 4. $F_4 = \{\mathbf{x}^{(3)}, \mathbf{x}^{(5)}\}$

Berbeda dengan NSGA-II yang menggunakan *crowding distance* untuk mempertahankan keragaman individu dalam populasi (Deb dkk., 2002), NSGA-III mengandalkan sekumpulan titik referensi (*reference points*). Titik-titik referensi ini dapat disesuaikan dengan karakteristik masalah yang diselesaikan. Jika tidak ada informasi mengenai preferensi solusi dari pengambil keputusan, titik-titik ini dapat ditentukan secara sistematis menggunakan metode yang disarankan oleh Das dan Dennis (1998). Titik-titik referensi dikonstruksi sekali di awal eksekusi NSGA-III

dan digunakan secara konsisten di setiap generasi sebagai tolok ukur keragaman populasi.

3.7.1 Penentuan Titik Referensi pada Hiperbidang

3.7.2 Inisialisasi Populasi

3.7.3 *Crossover* Individu dan Mutasi *Offspring* dalam Populasi

3.7.4 Pengurutan Takterdominasi Terhadap Populasi

3.7.5 Normalisasi Populasi

3.7.6 Asosiasi Individu dengan Titik Referensi

3.7.7 Preservasi *Niche*

3.7.8 Kompleksitas Waktu NSGA-III Per Generasi

3.8 CloudSim Plus

3.9 Pemrograman Linier untuk Masalah Penempatan VM dan Penentuan Rute

3.9.1 Pemrograman Nonlinier

Berikut disajikan model optimasi multiobjektif secara lengkap:

Tabel 3.3: Model Optimasi Lengkap untuk Penempatan VM dan Perutean Jaringan

Minimalkan	PC_{sum}	(O1)
Minimalkan	RW_{sum}	(O2)
Maksimalkan	BW_{sum}	(O3)
Minimalkan	D_{mean}	(O4)
dengan syarat	$\sum_{j=1}^{N_P} x_{ij} = 1$	(V1)
	$\sum_{i=1}^{N_V} v_i^{\text{cpu}} x_{ij} \leq p_i^{\text{cpu}}$	(V2)

Bersambung di halaman selanjutnya

Tabel 3.3: Model Optimasi Lengkap untuk Penempatan VM dan Perutean Jaringan (Lanjutan)

$$\sum_{i=1}^{N_V} v_i^{\text{mem}} x_{ij} \leq p_i^{\text{mem}} \tag{V3}$$

$$y_j = \begin{cases} 0 & \text{, jika } \sum_{i=1}^{N_V} x_{ij} = 0 \\ 1 & \text{, jika } \sum_{i=1}^{N_V} x_{ij} > 0 \end{cases} \tag{V4}$$

$$x_{ij} \in \{0, 1\} \tag{V5}$$

$$b_r \geq \frac{x_r}{M} \tag{R1}$$

$$b_r \leq x_r \cdot b_{\text{PM}}(p_i, p_j) \tag{R2}$$

$$\sum_{r \in \mathcal{R}^*(p_i, p_j)} x_r \leq k \tag{R3}$$

$$\sum_{(p_i,p_j) \in P \times P} \sum_{r \in \mathcal{R}^*(p_i,p_j) \,:\, l \in \text{range}(r)} b_r \leq c_l \tag{R4}$$

$$b_r \in \mathbb{R} \tag{R5}$$

$$x_r \in \{0, 1\} \tag{R6}$$

Fungsi objektif (O1), (O2), dan (O4) berturut-turut dapat dijabarkan menggunakan variabel keputusan sebagai berikut

$$\text{PC}_{\text{sum}} = \sum_{j=1}^{N_P} y_j \left((\text{PC}_j^{\max} - \text{PC}_j^{\text{idle}}) \sum_{i=1}^{N_V} \frac{x_{ij} v_i^{\text{cpu}}}{p_j^{\text{cpu}}} + \text{PC}_j^{\text{idle}} \right) \tag{O1}$$

$$\text{RW}_{\text{sum}} = \sum_{j=1}^{N_P} y_j \cdot \frac{\left| \sum_{i=1}^{N_V} x_{ij} (v_i^{\text{cpu}} p_j^{\text{mem}} - v_i^{\text{mem}} p_j^{\text{cpu}}) \right| + p_j^{\text{cpu}} p_j^{\text{mem}} \epsilon}{\sum_{i=1}^{N_V} x_{ij} (v_i^{\text{cpu}} p_j^{\text{mem}} + v_i^{\text{mem}} p_j^{\text{cpu}})} \tag{O2}$$

$$\text{D}_{\text{mean}} = \frac{\sum_{(p_i,p_j) \in P \times P} \sum_{r \in \mathcal{R}^*(p_i,p_j)} \Delta_r \cdot b_r}{\sum_{(p_i,p_j) \in P \times P} \sum_{r \in \mathcal{R}^*(p_i,p_j)} x_r} \tag{O4}$$

Perhatikan bahwa terdapat empat ekspresi yang belum berbentuk linier: fungsi

objektif (O1), (O2), (O4), dan kendala (V4). Hal tersebut dapat terlihat dari adanya perkalian antar variabel keputusan pada objektif (O1) dan (O2), nilai mutlak pada objektif (O2), bentuk pecahan pada objektif (O2) dan (O4), serta kondisi logika pada kendala (V4). Agar model di atas dapat diselesaikan menggunakan LP (*linear programming*) *solver*, model tersebut harus diubah ke dalam MILP (*Mixed Integer Linear Programming*). Langkah-langkah konversi setiap ekspresi akan di bahas pada bagian di bawah ini.

3.9.2 Konversi ke dalam Pemrograman Linier

Fungsi Objektif (O1) Untuk mempermudah pembahasan, fungsi (O1) dapat dinyatakan sebagai berikut:

$$\sum_{j=1}^{N_P} \sum_{i=1}^{N_V} P_{ij} x_{ij} y_j + \text{PC}_j^{\text{idle}}$$

di mana

$$P_{ij} = \frac{v_i^{\text{cpu}} (\text{PC}_j^{\text{max}} - \text{PC}_j^{\text{idle}})}{p_j^{\text{cpu}}}$$

Karena kebutuhan sumber daya setiap VM dan kapasitas sumber daya di setiap PM diasumsikan selalu tetap, P_{ij} dianggap sebagai konstanta. Perhatikan bahwa pada ekspresi di atas, terdapat perkalian antara variabel biner x_{ij} dan y_j, sehingga ekspresi tersebut tidak bersifat linier. Oleh karena itu, didefinisikan $N_V \cdot N_P$ buah variabel keputusan bantu $w_{ij} = x_{ij} y_j$. Kemudian, beberapa batasan untuk w_{ij} ditambahkan ke dalam model di atas sehingga diperoleh model berikut:

$$\begin{aligned}
\text{Minimalkan} \quad & \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} P_{ij} w_{ij} + \text{PC}_j^{\text{idle}} && \text{(O1a)} \\
\text{dengan syarat} \quad & w_{ij} \leq x_{ij} && \text{(W1)} \\
& w_{ij} \leq y_j && \text{(W2)} \\
& w_{ij} \geq x_{ij} + y_j - 1 && \text{(W3)} \\
& w_{ij} \in \{0, 1\} && \text{(W4)}
\end{aligned}$$

Fungsi Objektif (O2) Untuk mempermudah pembahasan, fungsi (O2) dapat dinyatakan sebagai berikut:

$$\sum_{j=1}^{N_P} y_j \cdot \frac{\left|\sum_{i=1}^{N_V} A_{ij}x_{ij}\right| + C_j}{\sum_{i=1}^{N_V} B_{ij}x_{ij}}$$

di mana

- $A_{ij} = v_i^{\text{cpu}} p_j^{\text{mem}} - v_i^{\text{mem}} p_j^{\text{cpu}}$,
- $B_{ij} = v_i^{\text{cpu}} p_j^{\text{mem}} + v_i^{\text{mem}} p_j^{\text{cpu}}$, dan
- $C_j = p_j^{\text{cpu}} p_j^{\text{mem}} \epsilon$

Karena kebutuhan sumber daya setiap VM dan kapasitas sumber daya di setiap PM diasumsikan selalu tetap, A_{ij}, B_{ij}, dan C_j dianggap sebagai konstanta.

Menghilangkan Nilai Mutlak Definisikan N_P buah variabel keputusan bantu $z_j = \left|\sum_{i=1}^{N_V} A_{ij}x_{ij}\right|$. Kemudian, tambahkan pula kendala-kendala baru untuk z_j ke dalam model multiobjektif di atas. Dengan demikian, diperoleh model baru sebagai berikut:

Tabel 3.4:

Minimalkan	$\sum_{j=1}^{N_P} \frac{z_j \cdot y_j + C_j \cdot y_j}{\sum_{i=1}^{N_V} B_{ij} \cdot x_{ij}}$	(O2a)
dengan syarat	$z_j \geq \sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z1)
	$z_j \geq -\sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z2)
	$z_j \geq 0$	(Z3)

Menghilangkan Perkalian Antar Variabel Keputusan Perhatikan bahwa pada pembilang fungsi objektif (O2a), terdapat perkalian dua variabel keputusan z_j dan y_j. Hal ini menyebabkan pembilang tersebut tidak bersifat linier. Linierisasi dapat dilakukan dengan mendefinisikan N_P buah variabel keputusan bantu $\alpha_j = z_j y_j$ dan ditambahkan pula kendala-kendala baru untuk ke dalam model di atas menggunakan metode linierisasi. Dengan demikian, diperoleh model baru sebagai berikut:

Tabel 3.5:

Minimalkan	$\sum_{j=1}^{N_P} \frac{\alpha_j + C_j \cdot y_j}{\sum_{i=1}^{N_V} B_{ij} \cdot x_{ij}}$	(O2b)
dengan syarat	$z_j \geq \sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z1)
	$z_j \geq -\sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z2)
	$\alpha_j \leq M y_j$	(A1)
	$\alpha_j \leq z_j$	(A2)
	$\alpha_j \geq z_j - M(1 - y_j)$	(A3)
	$\alpha_j \geq 0$	(A4)

Menghilangkan Pecahan Definisikan N_P buah variabel keputusan bantu $\beta_j = \frac{\alpha_j + C_j \cdot y_j}{\sum_{i=1}^{N_V} B_{ij} \cdot x_{ij}}$. Dengan menyubstitusi pecahan pada (O2b) dengan β_j, diperoleh model baru berikut

Tabel 3.6:

Minimalkan	$\sum_{j=1}^{N_P} \beta_j$	(O2c)
dengan syarat	$\beta_j \cdot (\sum_{i=1}^{N_V} B_{ij} \cdot x_{ij}) = \alpha_j + C_j \cdot y_j$	(B1)
	$z_j \geq \sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z1)
	$z_j \geq -\sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z2)
	$\alpha_j \leq M y_j$	(A1)
	$\alpha_j \leq z_j$	(A2)
	$\alpha_j \geq z_j - M(1 - y_j)$	(A3)
	$\alpha_j \geq 0$	(A4)

Perhatikan bahwa terdapat perkalian variabel keputusan β_j dengan x_{ij} pada kendala (B1). Linierisasi dapat diaplikasikan terhadap kendala (B1) untuk menghilangkan perkalian tersebut. Hal ini dilakukan dengan mendefinisikan $N_V \cdot N_P$ buah variabel keputusan bantu $\gamma_{ij} = x_{ij}\beta_j$ dan mengubah model di atas menjadi model berikut :

Tabel 3.7:

Minimalkan	$\sum_{j=1}^{N_P} \beta_j$	(O2c)
dengan syarat	$\sum_{i=1}^{N_V} B_{ij} \cdot \gamma_{ij} = \alpha_j + C_j \cdot y_j$	(G1)
	$\gamma_{ij} \leq M x_{ij}$	(G2)
	$\gamma_{ij} \leq \beta_j$	(G3)
	$\gamma_{ij} \geq \beta_j - M(1 - x_{ij})$	(G4)
	$\gamma_{ij} \geq 0$	(G5)
	$z_j \geq \sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z1)
	$z_j \geq -\sum_{i=1}^{N_V} A_{ij} \cdot x_{ij}$	(Z2)
	$\alpha_j \leq M y_j$	(A1)
	$\alpha_j \leq z_j$	(A2)
	$\alpha_j \geq z_j - M(1 - y_j)$	(A3)
	$\alpha_j \geq 0$	(A4)

Fungsi Objektif (O4) Perhatikan bahwa fungsi objektif (O4) dinyatakan sebagai pecahan dengan kombinasi linier variabel keputusan biner dan kontinu sebagai pembilang dan penyebutnya. Model pemrograman dengan fungsi objektif semacam ini dikategorikan sebagai *Mixed-Integer Linear Fractional Programming* (MILFP). Terdapat beberapa algoritma untuk menyelesaikan MILFP, salah satunya adalah algoritma Dinkelbach (You, Castro Grossman, 2009).

Akan tetapi, sebelum algoritma Dinkelbach dapat dijalankan, objektif (O4) harus dinyatakan sebagai masalah maksimalisasi terlebih dahulu. Perhatikan bahwa $\mathrm{D}_{\text{mean}} \geq 0$, sehingga

$$\frac{1}{\min \mathrm{D}_{\text{mean}}} = \max \frac{1}{\mathrm{D}_{\text{mean}}}$$

dan

$$\operatorname{argmin} \mathrm{D}_{\text{mean}} = \operatorname{argmax} \frac{1}{\mathrm{D}_{\text{mean}}}$$

Dengan demikian, MILFP akan mencari solusi yang memaksimalkan $1/\mathrm{D}_{\text{mean}}$, tetapi keoptimalan solusi tetap dihitung berdasarkan nilai D_{mean}.

Untuk mempermudah pembahasan, definisikan

$$Q(\mathbf{b}, \mathbf{x}) := \frac{1}{\mathbf{D}_{\text{mean}}} = \frac{A(\mathbf{b}, \mathbf{x})}{B(\mathbf{b}, \mathbf{x})}$$

di mana

- $\mathbf{b}$ adalah vektor keputusan kontinu dengan elemen b_r
- $\mathbf{x}$ adalah vektor keputusan biner dengan elemen x_r
- $A(\mathbf{b}, \mathbf{x}) = \sum \sum x_r$
- $B(\mathbf{b}, \mathbf{x}) = \sum \sum \Delta_r \cdot b_r$

Berikut algoritma Dinkelbach untuk mencari $\max Q(\mathbf{b}, \mathbf{x})$:

1. Definisikan $q_0 = 0$ dan inisialiasi $k \leftarrow 0$
2. Tentukan nilai $\mathbf{b}, \mathbf{x}$ yang memaksimalkan $A(\mathbf{b}, \mathbf{x}) - q_k B(\mathbf{b}, \mathbf{x})$ dengan syarat atau daerah keputusan yang sama dengan model awal. Misalkan solusi optimal untuk $\mathbf{b}$ dan $\mathbf{x}$ berturut-turut sebagai $\mathbf{b}_k$ dan $\mathbf{x}_k$
3. Jika $A(\mathbf{b}_k, \mathbf{x}_k) - q_k B(\mathbf{b}_k, \mathbf{x}_k) = 0$, hentikan algoritma dan keluarkan $\mathbf{b}_k$ dan $\mathbf{x}_k$ sebagai solusi optimal. Jika sebaliknya, definsikan $q_{k+1} = Q(\mathbf{b}_k, \mathbf{x}_k)$,tetapkan $k \leftarrow k + 1$, dan ulangi langkah kedua.

Algoritma Dinkelbach memanfaatkan fakta bahwa:

- $q^* = \max Q(\mathbf{b}, \mathbf{x}) = \max \frac{A(\mathbf{b}, \mathbf{x})}{B(\mathbf{b}, \mathbf{x})}$ jika dan hanya jika $F(q^*) = \max A(\mathbf{b}, \mathbf{x}) - q^* B(\mathbf{b}, \mathbf{x}) = 0$.
- Barisan $F(q_0), F(q_1), \ldots$ merupakan barisan bilangan nonnegatif menurun dengan laju konvergensi superlinier

Dengan demikian, $\mathbf{b}^*, \mathbf{x}^*$ memaksimalkan $Q(\mathbf{b}, \mathbf{x})$ ketika $A(\mathbf{b}^*, \mathbf{x}^*) - Q(\mathbf{b}^*, \mathbf{x}^*) B(\mathbf{b}^*, \mathbf{x}^*) = 0$.

Kendala (V4) Kendala (V4) menangani kasus *if-then* di mana sebuah PM hanya diaktifkan apabila terdapat VM yang menempatinya. Sejumlah kendala baru didefinisikan untuk menggantikan kasus *if-then* tersebut, yaitu

Tabel 3.8:

$x_{ij} \leq y_j$	(V4a)	untuk setiap $i = 1, \ldots, N_V$ dan $j = 1, \ldots, N_P$
$\sum_{i=1}^{N_V} x_{ij} \leq N_V \cdot y_j$	(V4b)	untuk setiap $j = 1, \ldots, N_P$

3.9.3 Pemrograman Linier

Meskipun LP dapat menggunakan lebih dari satu objektif, LP *solver* tidak bisa digunakan untuk menyelesaikan masalah multiobjektif. Oleh karena itu, solusi optimal Pareto dapat dicari menggunakan metode penjumlahan berbobot (*weighted sum*), seperti yang telah dibahas pada subbab Masalah Optimasi Multiobjektif.

Karena model yang dirumuskan di awal masih mengombinasikan masalah maksimalisasi dengan minimalisasi, setiap objektif harus diseragamkan. Mengingat bahwa $\max -f = -\min f$ dan $\operatorname{argmax} -f = \operatorname{argmin} f$, solusi optimal dapat diperoleh dengan mengubah semua masalah minimalisasi menjadi masalah maksimalisasi.

Dengan demikian, dalam perumusan MILP ini, objektif (O1) dan (O2) diubah menjadi masalah maksimalisasi $-\text{PC}_{\text{sum}}$ dan $-\text{RW}_{\text{sum}}$. Akan tetapi, keoptimalan solusi tetap dihitung menggunakan objektif asli, yaitu PC_{sum} dan RW_{sum}.

Berikut formulasi MILP untuk masalah multiobjektif penempatan VM dan penentuan rute

Tabel 3.9:

Maksimalkan	$w_1 \cdot (-\text{PC}_{\text{sum}}) + w_2 \cdot (-\text{RW}_{\text{sum}}) + w_3 \cdot \text{BW}_{\text{sum}} + w_4 \cdot F$	(O)
dengan syarat	$\text{PC}_{\text{sum}} = \sum_{j=1}^{N_P} \sum_{i=1}^{N_V} P_{ij} w_{ij} + \text{PC}_j^{\text{idle}}$	
	$\text{RW}_{\text{sum}} = \sum_{j=1}^{N_P} \beta_j$	
	$\text{BW}_{\text{sum}} = \sum \sum b_r$	
	$F = A(\mathbf{b}, \mathbf{x}) - q_k B(\mathbf{b}, \mathbf{x})$	
	$A(\mathbf{b}, \mathbf{x}) = \sum \sum x_r$	

Bersambung di halaman selanjutnya

Tabel 3.9: (Lanjutan)

$B(\mathbf{b}, \mathbf{x}) = \sum \sum \Delta_r \cdot b_r$	
(V1)-(V4)	
(V5a)-(V5b)	
(N1)-(N6)	
(W1)-(W3)	
(G1)-(G5)	
(Z1)-(Z2)	
(A1)-(A4)	

MILP ini menggunakan beberapa variabel keputusan: x_{ij}, y_j, x_r, w_{ij} sebagai variabel keputusan biner, dan $b_r, \gamma_{ij}, z_j, \alpha_j$ sebagai variabel keputusan kontinu nonnegatif.

Seperti yang dibahas pada bagian mengenai konversi objektif (O4) melalui algoritma Dinkelbach, MILP ini akan diselesaikan oleh *solver* berkali-kali menggunakan nilai q_k yang berbeda hingga $\max_{\mathbf{b},\mathbf{x}} F(q_k) = 0$.

3.10 DOCPLEX

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

4.1 Deskripsi Umum Sistem

4.2 Analisis Kebutuhan Sistem

4.3 Pembuatan Sistem

4.3.1 Pembuatan Sistem Pengenalan Entitas Bernama

4.3.2 Pembuatan Sistem Ekstraksi Kalimat Pernyataan

4.4 Rancangan Antarmuka

4.4.1 Deskripsi

4.4.2 *Wireframe*

Lorem ipsum odor amet, consectetuer adipiscing elit. Cursus viverra fames inceptos neque imperdiet nostra duis. Dignissim arcu at tempor mattis curae sed nascetur aliquet luctus. Netus arcu venenatis semper suscipit consequat. Phasellus congue sodales blandit ultricies donec dignissim. Dapibus at odio penatibus mauris adipiscing fusce sodales. Quisque nullam massa ullamcorper curae neque vehicula ultricies. Primis bibendum etiam velit viverra arcu etiam sed malesuada ut.

Vulputate ad malesuada elementum et mollis parturient sodales. Netus lectus vitae sit risus netus ipsum congue diam. Faucibus nascetur malesuada risus luctus ridiculus. Suspendisse nec ridiculus accumsan justo parturient metus iaculis. Montes nulla ultricies fringilla nascetur nisi dignissim massa lectus sagittis. Mi tellus orci nullam etiam scelerisque pretium inceptos id feugiat. Lacus luctus natoque placerat cursus faucibus. Luctus porta eget orci nullam magna nostra viverra eget.

Aptent accumsan ac torquent nibh magna tincidunt facilisis facilisi. Libero quis dignissim rhoncus aptent sapien faucibus nostra. Hendrerit volutpat faucibus diam sollicitudin aliquet diam lacus. Hac sed est dictum felis lacus congue at potenti. Metus sollicitudin varius suspendisse consequat scelerisque curae. Luctus porttitor cursus vel neque ipsum egestas. At orci sagittis pulvinar curabitur; ipsum adipiscing nullam diam. Pulvinar euismod interdum aliquam commodo augue aliquam erat.

Facilisi dictum imperdiet elit arcu erat dignissim neque. Hac tristique potenti; curabitur fusce aenean leo.

Diam euismod facilisis libero in sem. Ad et justo morbi vel justo primis ipsum cras et? Fermentum lacinia faucibus tristique pharetra fringilla ad. Eu ut integer consequat odio molestie. Nisl lectus ornare erat primis amet laoreet ultricies ligula consequat. Nibh tristique integer iaculis eget phasellus est magna. Fames risus rhoncus turpis sem ad netus massa efficitur.

Fames litora imperdiet accumsan nascetur nam arcu cursus. Odio vel sed platea tempor aptent senectus, consectetur conubia. Leo aenean vitae ultrices quis proin sit. Litora dictum torquent interdum morbi velit adipiscing. Nostra pharetra facilisi iaculis bibendum taciti quisque erat. Justo phasellus sed massa convallis turpis magnis facilisis. Dignissim libero sapien phasellus hendrerit ultricies. Adipiscing faucibus sodales justo hendrerit sagittis imperdiet felis maximus.

BAB V

IMPLEMENTASI SISTEM

5.1 Spesifikasi

5.2 Implementasi Sistem Pengenalan Entitas Bernama

5.3 Implementasi Sistem Ekstraksi Kalimat Pernyataan

BAB VI

PENGUJIAN DAN PEMBAHASAN SISTEM

6.1 Pengujian Sistem Pengenalan Entitas Bernama

6.2 Pengujian Sistem Ekstraksi Kalimat Pernyataan

BAB VII

PENUTUP

7.1 Kesimpulan

7.2 Saran

DAFTAR PUSTAKA

Crockford, Douglas., 2006, *The application/json media type for javascript object notation (json)*.

LAMPIRAN A

BERKAS JSON UNTUK MODEL SISTEM PENGENALAN ENTITAS BERNAMA