JURNAL REKAYASA SISTEM DAN INDUSTRI
e-ISSN: 2579-9142 p-ISSN: 2356-0843
http://jrsi.sie.telkomuniversity.ac.id/index.php/JRSI

Perancangan Aplikasi untuk Perbandingan Algoritma Penjadwalan dengan Mempertimbangkan Proses Real Berbasis Event Logs

Application Design for Comparison of Job Scheduling Algorithm by Condisering Real Process Based on Event Logs

Nur Ichsan Utama*[1], Rio Aurachman[2]
Program Studi Sistem Informasi, Fakultas Rekayasa Industri, Telkom University
Program Studi Sistem Informasi, Fakultas Rekayasa Industri, Telkom University

ARTICLE INFO

Article history:
Diterima xx-xx-xx
Diperbaiki xx-xx-xx
Disetujui xx-xx-xx

Kata Kunci:
Algoritma Penjadwalan, *Process mining*, Simulation, Event Log, Petri Net

ABSTRAK

Kondisi *job* yang ditemukan di lapangan pada umumnya lebih menyerupai kondisi yang terjadi pada kasus penjadwalan *dynamic job shop* dibandingkan dengan *static job shop*. Pada paper ini, penulis mencoba merancang sebuah software aplikasi untuk membandingkan performansi algoritma penjadwalan dengan mempertimbangkan kondisi *job* yang dinamis dan *uncertain* berdasarkan data *event log*. Data *event log* digunakan untuk menghasilkan *process model* yang dapat merepresentasikan kondisi real proses menggunakan teknik *process mining*. Untuk lebih memodelkan kondisi nyata yang terjadi di lapangan, penulis membuat agar waktu proses pengerjaan *operation* dapat terjadi dengan waktu yang *uncertain*. Waktu proses yang *uncertain* adalah kondisi dimana waktu proses pengerjaan tidak dapat dipastikan untuk selesai dalam durasi waktu tertentu. Untuk memverifikasi dan membandingkan performansi algoritma penjadwalan, kami akan mengkombinasikan hasil *process model*, distribusi waktu *operation*, dan kondisi *resource* berdasarkan dari data *event log* dengan mekanisme simulasi. Indikator performansi yang dapat dibandingkan dan dianalisa meliputi *throughput*, *flow time*, *waiting time*, dan makespan. Colored Petri Net dan Gantt Chart akan digunakan untuk merepresentasikan dalam bentuk visual simulasi yang sedang berjalan secara real time sehingga lebih memudahkan dalam menganalisis angka indikator performansi. Aplikasi ini dirancang untuk dijalankan online agar siapa saja dapat mengunggah dan membandingkan algoritma penjadwaannya dengan algoritma lain yang sudah pernah diunggah sebelumnya pada kasus event log tertentu.

Keywords:
Job Scheduling Algorithm, Process mining, Simulation, Event Log, Petri Net

ABSTRACT

In the real-world manufacturing plant, usually, job happens more similar with dynamic job shop scheduling rather than with static job shop scheduling. In this paper, author attempt to design an application to compare the performance of job scheduling algorithm considering condition of job which is dynamic and uncertain based on event log data. Event log data is used to generate process model that will represent condition of the real process using process mining technique. We consider processing time of operation is uncertain to better model what happened in the real condition. This uncertain processing time is represented imprecise values duration to finishing an operation of a job. We will combine the process model, operation time distribution, and resource condition based on the event log with simulation mechanism to verify and compare the performance of each scheduling algorithm. Several performance indicators that can be compared include throughput, flow time, waiting time., and makespan. Colored Petri Net and Gantt Chart will be used to represent on going simulation in real time to easier the user to analyze the performance indicator. The application is designed to be run online so that anyone can upload and compare the scheduling algorithm with other algorithms that have been uploaded previously for specific event logs. This comparison method is expected to be useful for the industry because it considers the real conditions in the field based on event log data.

*Penulis korespondensi
Email: nichsan@telkomuniversity.ac.id (Utama, N.I.), rioaurachman@telkomuniversity.ac.id (Aurachman, R..)

doi: https://doi.org/10.25124/jrsi.v5i02.299

1. Pendahuluan

Dalam dunia industri saat ini yang penuh dengan persaingan, perusahaan manufaktur dihadapkan pada tantangan untuk dapat menjaga *customer satisfaction*. Salah satu aspek yang berpengaruh cukup signifikan terhadap *customer satisfaction* adalah terjaminnya ketersediaan stok barang di pasar. Ketersediaan stok barang ini tidak dapat dilepaskan dari mekanisme produksi yang terjadi di *shop floor*. Mekanisme produksi yang tidak efektif dapat berkontribusi terhadap semakin lamanya waktu pengerjaan sebuah *job*. Perencanaan dan penjadwalan proses produksi adalah hal yang harus dipertimbangkan untuk dievaluasi dalam penerapan mekanisme produksi yang lebih efektif karena dapat meningkatkan utilisasi penggunaan sumber daya yang tersedia.

Sehubungan dengan penjadwalan pengerjaan *job* pada proses produksi banyak pilihan algoritma yang dapat diterapkan. Seringkali pilihan algoritma ini disesuaikan dengan kondisi *job* dan kondisi mesin yang ada pada *shop floor* di lapangan dan objektif tertentu yang ingin dicapai. Pada kasus dimana pola *job* yang ada di lapangan adalah *job shop* dan jumlah *job* yang diperhitungkan dalam penjadwalan minimal 3, sulit untuk mendapatkan solusi yang optimal dengan waktu komputasi yang terbatas dikarenakan kasus demikian sudah terbukti termasuk kategori NP-hard [1]. Untuk menyelesaikan kasus demikian maka sebagian peneliti memilih untuk menggunakan algoritma metaheuristic yang sudah terbukti dapat memberikan hasil yang cukup baik dan bahkan terkadang optimal dengan waktu komputasi yang cukup singkat.

Kondisi *job* yang ditemukan pada perusahaan manufaktur pada umumnya lebih menyerupai kondisi yang terjadi pada kasus penjadwalan *dynamic job shop* dibandingkan dengan kasus penjadwalan *static job shop*. Pada kasus penjadwalan *dynamic job shop*, penjadwalan ulang perlu dilaksanakan pada kondisi waktu yang tidak pasti (*random*) sementara mesin tetap terus bekerja menyelesaikan *job* yang sudah masuk dan sedang diproses saat itu. Kasus penjadwalan ulang ini dapat disebabkan oleh beberapa sebab diantaranya dikarenakan job baru yang datang dan terjadinya gangguan yang menyebabkan berhentinya proses produksi pada mesin tertentu. Pada paper ini, kondisi *dynamic job shop* yang akan dipertimbangkan adalah kondisi dimana *job* baru yang terus datang secara kontinyu dalam interval waktu tertentu. Untuk lebih memodelkan kondisi nyata yang terjadi di lapangan, kami mempertimbangkan kondisi yang memungkinkan waktu proses pengerjaan operation dari sebuah *job* di mesin terjadi dalam waktu proses yang *uncertain*. Waktu proses yang *uncertain* dalam hal ini adalah kondisi dimana waktu proses pengerjaan tidak dapat dipastikan untuk dapat selesai dalam durasi waktu yang pasti. *Fuzzy number* akan digunakan untuk merepresentasikan kondisi waktu proses yang *uncertain* ini.

Hasil implementasi tahap awal yang dijelaskan pada bagian akhir paper ini tidak merepresentasikan rancangan aplikasi berdasarkan paper ini secara keseluruhan. Hal ini dikarenakan pada tahap ini belum semua fitur terimplementasi melainkan hanya fitur dasar saja yang diimplementasi meliputi fitur simulasi sederhana dan penerapan algoritma penjadwalannya. Pada *paper* ini, kami mempertimbangkan waktu proses operasi di sebuah mesin tidak dapat ditentukan secara pasti sehingga dalam proses simulasinya kami menggunakan model distribusi tertentu untuk merepresentasikan waktu yang tidak pasti. Untuk tahap awal ini, kami menggunakan distribusi sederhana untuk merepresentasikan ketidakpastian tadi yaitu probabilitas distribusi triangular. Aplikasi yang akan dibuat berdasarkan rancangan di paper ini akan dibuat menggunakan bahasa pemrograman java. Untuk tahap awal kami mencoba menerapkan aplikasi destop menggunakan fitur *Library* JavaFx yang tersedia di Java yang dapat digunakan untuk membuat implementasi visual grafik yang cukup rumit dan fleksibel.

2. Studi Literatur

Banyak metode yang telah diusulkan oleh para peneliti dalam *paper* penelitian untuk menyelesaikan kasus penjadwalan *dynamic job shop* atau dynamic job shop scheduling (DJSS). Lin, salah satu diantara peniliti, mencoba untuk menerapkan algoritma metaheuristic genetic algorithm pada kasus DJSS dan mengkomparasi hasilnya berdasarkan berbagai kasus objektif yang berbeda-beda [2]. Adibi mencoba mengkombinasikan algoritma metaheuristic variable neighborhood search dengan artifical neural network untuk meminimasi *mean flow time* untuk kasus DJSS dengan mmpertimbangkan kondisi *job* baru yang datang terus menerus dan breakdown mesin [7]. Sharabi menggunakan algoritma reinforcement learning untuk mengestimasi parameter yang tepat bagi algoritma variable neighborhood search untuk kasus DJSS [15].

Dalam kasus lain para peneliti mencoba mempertimbangkan beberapa aspek yang ada pada sebuah *job* tidak dapat ditentukan secara pasti (*uncertain*). Sakawa mencoba mempertimbangkan *fuzzy processing time* dan fuzzy duetime dengan objektif untuk meminimasi makespan dan *maximum average agreement index* namun untuk kasus static job shop atau static job shop scheduling (SJSS) [3]. *Fuzzy processing time* disini adalah kondisi dimana waktu penyelesaian dari sebuah *operation* di sebuah *job* tidak dapat ditentukan secara pasti. Niu juga mencoba mempertimbangkan *fuzzy processing time* untuk digunakan pada algoritma metaheuristic particle swarm optimization yang dikombinasikan dengan genetic operator pada kasus SJSS [4]. Jamrus juga mencoba mempertimbangkan kondisi waktu proses yang tidak pasti namun untuk kasus SJSS yang lebih spesifik yaitu pada manufaktur semikondutor [17]. Pada kasus yang lain beberapa peneliti mencoba mempertimbangkan kondisi yang tidak pasti pada sebuah *job* tadi pada DJSS. Liu mencoba menggunakan algoritma fast estimation of distribution dengan mempertimbangkan *fuzzy process time* untuk meminimasi makespan [10]. Sharma mencoba membandingkan performansi beberapa metode dispatching rule pada kasus stokastik DJSS [12]. Stokastik DJSS pada dasarnya juga mempertimbangkan kondisi yang tidak pasti pada DJSS baik itu kondisi waktu tiba job, waktu proses di mesin, atau lainnya namun dikhususkan menggunakan metode probabilistik.

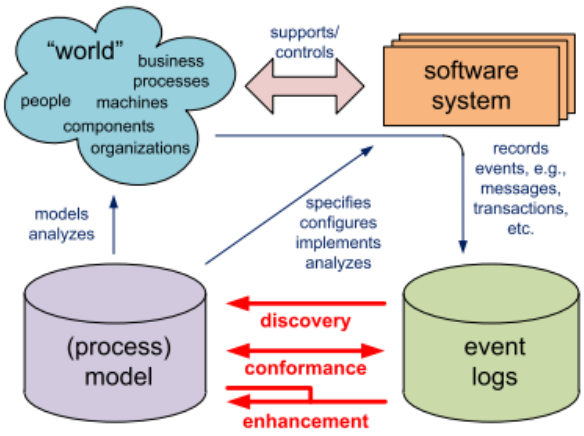
Perbedaan yang dilakukan pada penelitian di *paper* ini dibandingkan penelitian yang telah ada pada *paper-paper* sebelumnya adalah kami mencoba membuat sebuah aplikasi yang dapat digunakan secara mudah, fleksibel dan objektif untuk membandingkan performansi algoritma pada kasus DJSS dengan mempertimbangkan kondisi waktu proses yang tidak pasti (*uncertain*) dengan mempertimbangkan proses real dari data event log. Fitur simulasi yang terintegrasi dengan aplikasi akan memudahkan pengguna untuk melihat dan membandingkan performansi dari algoritma yang digunakan. Paper ini merupakan tahap awal perancangan aplikasi dari rencana selanjutnya pengembangan aplikasi sesuai dengan rancangan yang kami buat pada paper ini.

2.1 Process mining

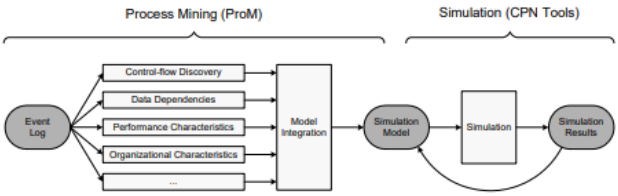
Process mining merupakan teknologi yang memanfaatkan event log untuk menganalisa alur kerja [8]. Hasil output dari *process mining* dapat dimanfaatkan untuk memberikan tanda kemungkinan masalah yang mungkin terjadi dalam sebuah model alur kerja seperti bottleneck yang mana kondisi ini sulit untuk diidentifikasi dengan hanya menggunakan model statik.

Pada awalnya *process mining* lebih terfokus untuk menghasilkan sebuah alur process saja dari data event log (*discover process*). Namun dalam perkembangannya, cakupan *process mining* menjadi semakin luas meliputi *conformance checking*, *operational support*, dan cakupan lainnya. Untuk menghasilkan sebuah alur proses dari data event log beberapa algoritma process discovery pada *process mining* dapat digunakan, diantaranya algoritma alpha dan inductive-miner. Kualitas alur proses yang dihasilkan oleh algoritma *process discovery process mining* sangat ditentukan oleh banyak faktor dan menentukan kualitasnya tidaklah mudah. Peneliti yang bekerja pada area *process mining* pada umumnya menggunakan empat kriteria kualitas yang saling mempengaruhi satu sama lain, yaitu fitness, simplicity, generalization, dan precision [11]. Dikarenakan dalam paper ini hasil dari alur proses akan digunakan untuk menjalankan simulasi, harus dipastikan bahwa alur proses yang dihasilkan tidak akan mengalami deadlock. Gambar 1 menunjukan tiga jenis *process mining* yang utama yaitu *discovery process model*, *conformance checking*, dan *enhancement*.

Penerapan *process mining* yang digunakan untuk menghasilkan sebuah model simulasi telah dilakukan oleh Rozinat [6]. Rozinat menggabungkan beberapa fitur *process mining* yang ada pada ProM framework yaitu control-flow discovery, data dependencies, performance characteristic, dan organizational characteristic seperti yang terlihat pada gambar 2 menjadi sebuah model simulasi. Berdasarkan eksperimen yang dilakukan penulis menggunakan aplikasi ProM, model simulasi yang dibuat oleh Rozinat [6] selalu menggunakan asumsi dispatching rule first-come-first-served untuk setiap job (token dalam konteks *process mining*) yang datang. Pada paper ini kami mencoba menambahkan fitur dimana pengerjaan sebuah job tidak selalu berdasarkan dispatching first-come-first-served namun dapat ditentukan berdasarkan algoritma penjadwalan yang dipilih. Dimana kemudian hasil dari simulasinya dapat digunakan untuk membandingankan mana algoritma penjadwalan yang lebih baik pada kondisi yang mendekati real berdasarkan data event log.



Gambar 1 *Process mining: discovery, conformance, ehancement* [11]

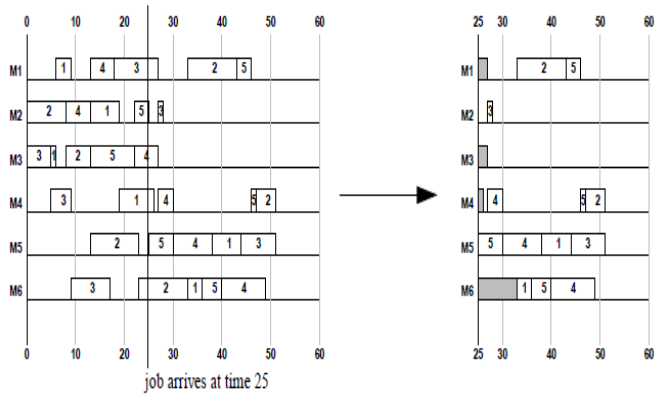


Gambar 2 Kombinasi teknik *process mining* untuk menghasilkan model simulasi [6]

2.2 Dynamic job shop scheduling

Penjadwalan *job shop* atau *job shop scheduling* secara umum terdiri atas sekumpulan *job* dan mesin. Setiap *job* masing-asing memiliki urutan proses operasi yang harus dilalui pada mesin, yang mana hal ini akan mendefinisikan *precedence constraint* [2]. Kasus penjadwalan untuk *job shop* secara umum biasa disebut dengan istilah *job shop scheduling problem* (JSSP). Berdasarkan waktu rilis *job*, JSSP dapat dibagi menjadi dua yaitu *static job shop scheduling problem* (SJSS) dan *dynamic job shop scheduling problem* (DJSS). Pada kasus SJSS semua *job* diasumsikan sudah siap pada waktu nol (sejak awal). Sedangkan pada kasus DJSS, waktu rilis sebuah *job* tidak ditentukan sebelumnya, yang mana pada kasus ini *job* datang di waktu yang berbeda-beda. DJSS juga dapat dibagi lebih lanjut ke dalam dua klasifikasi yaitu deterministik dan stokastik berdasarkan spesifikasi waktu rilis *job*. Pada kasus deterministik, meskipun waktu rilis job berbeda-beda namun sudah diketahui sejak awal, sedangkan untuk kasus stokastik, waktu rilis *job* merupakan variabel yang *random* yang dideskripsikan berdasarkan probabilitas distribusi tertentu [2]. Pada kasus DJSS dimana sebuah *job* dirilis pada waktu t, maka pada waktu t tersebut, penjadwalan yang sudah dihitung dan dibuat sebelumnya harus dikalkulasi ulang (*rescheduling*). Hal ini dilakukan untuk mendapatkan hasil yang lebih baik berdasarkan objektif yang ingin dicapai. Sebagai ilustrasi kasus rescheduling pada dilihat pada Gambar 2 di bawah ini. Gambar 2 pada bagian kiri menggambarkan hasil penjadwalan yang sudah dikalkulasi dan dibuat sebelum *job* dirilis pada waktu t=25. Ketika kemudian sebuah *job* datang pada wkatu t=25, maka semua operasi yang belum dijalankan pada waktu t >= 25 harus diikutkan dalam proses

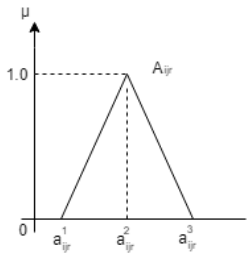
kalkulasi penjadwalan ulang. Sedangkan untuk operasi yang pada saat t=25 masih berjalan hanya dipertimbangkan kondisi sisa waktunya yang harus diselesaikan. Sisa waktu untuk tiap-tiap operasi yang masih berjalan ini haru dipertimbangkan pada kasus rescheduling agar didapatkan hasil penjadwalan yang merepresentasikan kondisi yang sebenarnya.



Gambar 3 Rescheduling pada waktu t = 25 dikarenakan job baru dirilis [2]

2.3 Dynamic job shop scheduling dengan kondisi uncertain processing time

Pada kasus job shop scheduling dengan kondisi *uncertain processing time*, teori fuzzy set dapat diimplementasikan disini [4]. Kondisi waktu proses yang tidak pasti dapat direpresentasikan dengan menggunakan triangular fuzzy number (TFN).



Gambar 4 Fuzzy processing time

Seperti yang terlihat pada Gambar 4, kondisi waktu proses yang tidak pasti dapat drepresentasikan dengan menggunakan TFN A_{ijr} dalam bentuk triplet $(a^1_{ijr}, a^2_{ijr}, a^3_{ijr})$. Untuk kasus dimana dua buah dua triangular fuzzy number A = (a^1, a^2, a^3) dan B = (b^1, b^2, b^3), maka penjumlahan dari A dan B dapat didefinisikan persamaan 1 sebagai berikut :[3]

$$A + B = (a^1, a^2, a^3) + (b^1, b^2, b^3) = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \quad (1)$$

Pada kasus waktu proses yang tidak pasti, kondisi yang dijelaskan sebelumnya pada kasu *rescheduling* seperti yang diilustrasikan pada Gambar 2 menjadi berbeda dan lebih sulit. Hal ini dikarenakan dua hal, waktu proses yang tidak pasti menyulitkan untuk membandingkan nilai objektif dari suatu solusi tertentu dan sulitnya untuk menentukan waktu sisa untuk operasi yang ketika *job* baru datang pada waktu t masih berjalan.

2.4 Model Simulasi untuk dynamic job shop scheduling

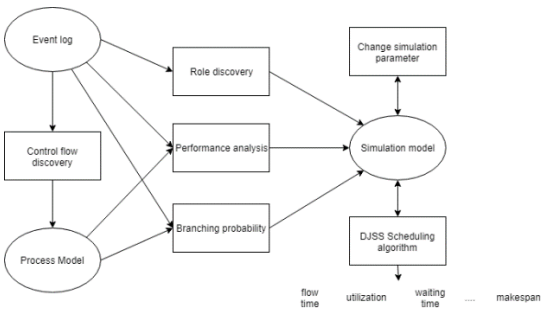
Literatur yang ada untuk model simulasi pada kasus DJSS meliputi simulasi yang berdasarkan pada kondisi aktual atau hipotesis [5]. Berkut adalah daftar asumsi model yang biasanya digunakan untuk simulasi [5]:

a. *Job* terdiri atas urutan operasi yang jelas.
b. Sebuah operasi dapat dijalankan pada satu mesin tertentu saja.
c. Hanya ada satu mesin untuk setiap tipe.
d. Waktu proses dan *due dates* diketahui pada saat ketika *job* tiba.
e. Setup time tidak dipengaruhi oleh urutan operasi.
f. Sekali operasi dijalankan tidak dapat dinterupsi.
g. Sebuah operasi tidak dapat dijalankan sebelum operasi yang menjadi *predecessor* nya selesai.
h. Setiap mesin dapat memproses hanya satu operasi dalam satu waktu.
i. Setiap mesin secara kontinyu selalu tersedia untuk proses produksi.

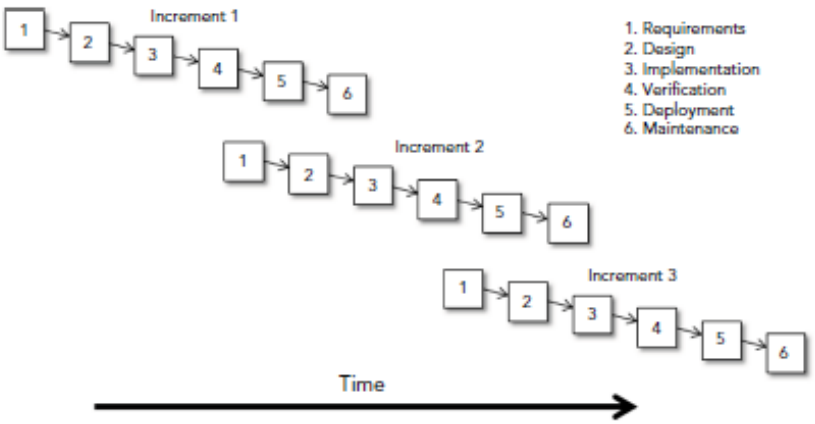
Untuk tahap awal implementasi di paper ini, kami menggunakan asumsi seperti yang tertulis diatas, namun rancangan kami selanjutnya akan menerapkan simulasi dengan kondisi yang lebih kompleks dengan mempertimbangkan real proses berbasis event log.

3. Metode Penelitian

Metodologi penelitian yang digunakan pada penelitian ini adalah metode *iterative model*. Metode iterative model sesuai digunakan untuk pembuatan aplikasi yang belum dapat diketahui kebutuhannya secara keseluruhan pada awalnya [13]. Aplikasi kemudian dikembangkan secara bertahap dengan penambahan atau perbaikan fitur tertentu. Metode ini dipilih dikarenakan memang sejak awal penulis berencana membuat aplikasi penjadwalan untuk kasus DJSS ini secara bertahap. Tujuan akhirnya agar aplikasi ini dapat digunakan oleh para peneliti yang melakukan riset seputar masalah penjadwalan DJSS dengan memperhitungkan kondisi proses yang real berdasarkan event log. Hasil pengembangan tahap awal seperti yang direncanakan adalah berupa aplikasi penjadwalan yang simpel untuk kasus DJSS dengan mempertimbangkan kondisi waktu yang tidak pasti yang sudah dilengkapi dengan fitur simulasi. Tahap-tahap dari iterative model dapat dilihat pada Gambar 6.



Gambar 5 Arsitektur rancangan aplikasi



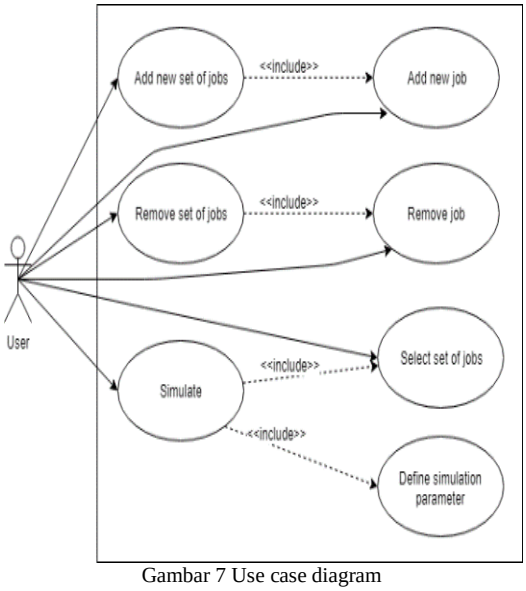
Gambar 6 Iterative model [13]

3.1 Use Case Diagram

Use case diagram merupakan diagram yang menggambarkan interaksi pengguna dengan system atau aplikasi [13]. Seperti yang terlihat pada Gambar 7, fitur yang tersedia pada aplikasi tahap awal ini masih terbatas pada pengaturan kumpulan *job* dan simulasi. Implementasi algoritma penjadwalan ada di dalam fitur simulasi, dimana algoritma penjadwalan akan bekerja me-*reschedule* setiap ada *job* baru yang dirilis pada waktu tertentu.

3.2 Class Diagram

Class diagram adalah diagram yang mendeskripsikan *classes* yang membangun system meliputi properti, fungsi, dan hubungan di antara *classes* itu sendiri. *Class diagram* dari aplikasi dapat dilihat pada Gambar 8. Berdasarkan struktur *class diagram* tersebut, fitur utama berupa simulasi ada di dalam class SimulateDJSS.



Gambar 7 Use case diagram

3.3 Entity Relationship Diagram

ERD merupakan diagram yang berguna untuk mendeskripsikan hubungan antar data yang ada pada basis data [14]. Pada pembentukan ERD akan ada 3 komponen yang akan dibentuk yaitu entitas, relasi, dan atribut. Sebuah tabel yang ada pada *database* mewakili sebuah entitas data. Aplikasi tahap awal yang dibuat pada paper ini hanya membutuhkan struktur *database* yang sangat sederhana dikarenakan penyimpanan data permanan hanya digunakan untuk menyimpan informasi kumpulan *job* yang akan digunakan pada kalkulasi penjadwalan dan simulasi. Gambar 9 menunjukan ERD yang digunakan pada aplikasi.

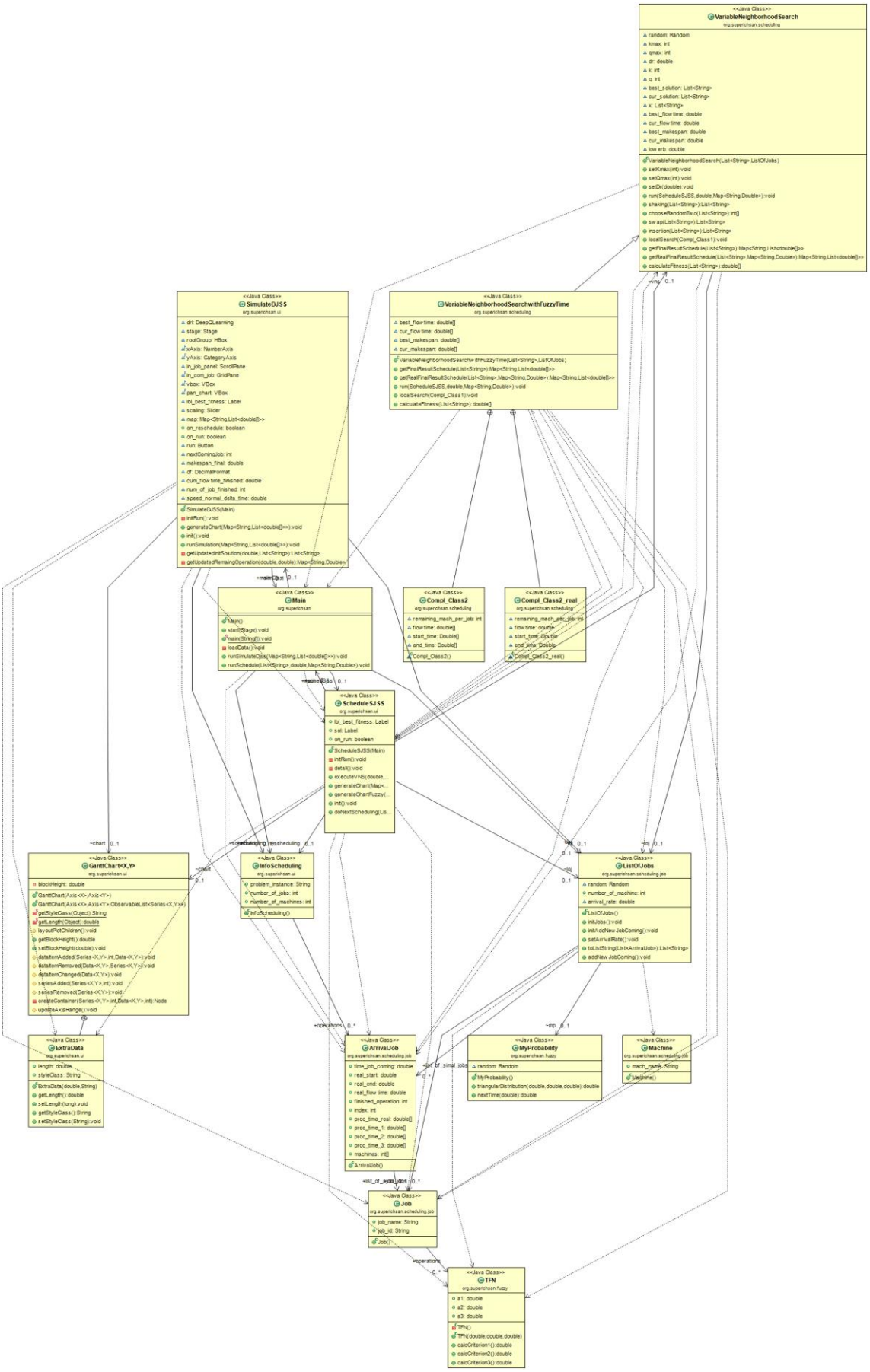
3.4 Konfigurasi dynamic job shop

Aplikasi secara *default* sudah menyimpan data *job* yang dapat digunakan oleh pengguna apabila tidak mau direpotkan dengan proses input data *job* baru. Data *default job* yang digunakan oleh aplikasi diambil dari data problem *job shop* yang sudah dikenal dan banyak digunakan oleh para peneliti lainnya yaitu abz6 seperti yang terlihat pada Gambar 10, yang terdiri atas 10 *job* dan 10 mesin. Untuk menyesuaikan dengan kondisi waktu proses yang *uncertain* yang digunakan pada aplikasi maka data abz6 ini dirubah sedemikian rupa sehingga setiap operasi memiliki tiga input waktu proses yaitu a1, a2, dan a3 dimana a2 diberi nilai berdasarkan waktu default dari problem abz6. Sedangkan untuk a1 dan a3 diset dengan nilai masing-masing berupa pengurangan dan penambahan dari waktu a2 dengan delta bervariasi dari 1-20.

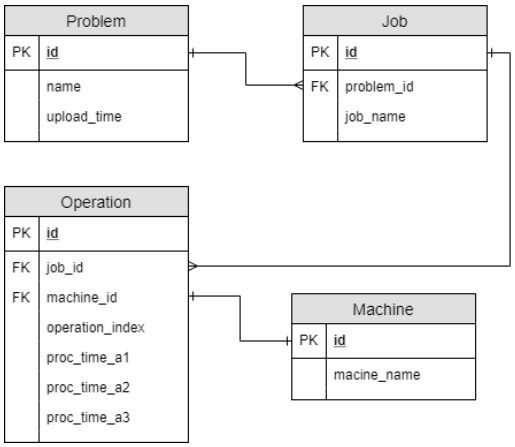
3.5 Struktur model simulasi

Model simulasi yang diterapkan pada aplikasi ini sedikit berbeda dengan model simulasi yang dijelaskan pada studi literatur. Berikut adalah daftar asumsi model simulasi yang digunakan pada aplikasi:

a. *Job* terdiri atas urutan operasi yang jelas.
b. Sebuah operasi dapat dijalankan pada satu mesin tertentu saja.
c. Hanya ada satu mesin untuk setiap tipe.
d. Waktu proses tidak diketahui pada saat ketika *job* tiba.
e. Setup time tidak dipengaruhi oleh urutan operasi.
f. Sekali operasi dijalankan tidak dapat dinterupsi.
g. Sebuah operasi tidak dapat dijalankan sebelum operasi yang menjadi *predecessor* nya selesai.
h. Setiap mesin dapat memproses hanya satu operasi dalam satu waktu.
i. Setiap mesin secara kontinyu selalu tersedia untuk proses produksi.



Gambar 8 Class Diagram



Gambar 9 ERD aplikasi

```
instance abz6
++++++++++++++++++++++++++++
Adams, and Zawack 10x10 instance (Table 1, instance 6)
10 10
7 62 8 24 5 25 3 84 4 47 6 38 2 82 0 93 9 24 1 66
5 47 2 97 8 92 9 22 1 93 4 29 7 56 3 80 0 78 6 67
1 45 7 46 6 22 2 26 9 38 0 69 4 40 3 33 8 75 5 96
4 85 8 76 5 68 9 88 3 36 6 75 2 56 1 35 0 77 7 85
8 60 9 20 7 25 3 63 4 81 0 52 1 30 5 98 6 54 2 86
3 87 9 73 5 51 2 95 4 65 1 86 6 22 8 58 0 80 7 65
5 81 2 53 7 57 6 71 9 81 0 43 4 26 8 54 3 58 1 69
4 20 6 86 5 21 8 79 9 62 2 34 0 27 1 81 7 30 3 46
9 68 6 66 5 98 8 86 7 66 0 56 3 82 1 95 4 47 2 78
0 30 3 50 7 34 2 58 1 77 5 34 8 84 4 40 9 46 6 44
++++++++++++++++++++++++++++
```

Gambar 10 Dataset abz6

Untuk memodelkan kondisi stokastik pada simulasi berikut adalah detail distribusi yang digunakan pada aplikasi. Untuk memodelkan kedatangan sebuah *job* baru kami harus memperhitungkan *inter-arrival time* yaitu waktu rata-rata diantara kedatangan dua buah *job*. Berdasarkan literatur, *inter-arrival time* harus ditentukan untuk membuat utilisasi mesin < 100% [12]. Jika tidak, maka jumlah *job* yang ada di antrian menunggu di depan sebuah mesin akan terus bertambah tanpa batas [12]. Berdasarkan literatur, pada umumnya proses kedatangan sebuah *job* mengikuti distribusi possion [12]. Oleh karena itu, *inter-arrival time* akan mengikuti distribusi eksponensial. *Mean job arrival rate* dapat dikalkulasi menggunakan persamaan 2 sebagai berikut: [12]

$$\lambda = \frac{U \cdot M}{\mu_p \cdot \mu_g} \qquad (2)$$

λ = mean job arrival rate
μ_p = mean processing time per operation
μ_g = mean number of operation per job
U = shop utilization
M = number of machines in the shop

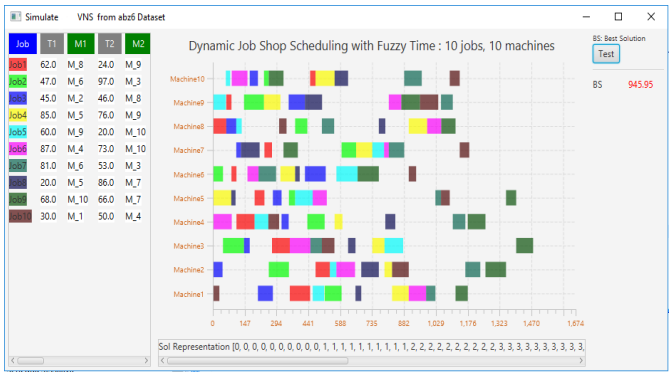
Pada paper ini, kami menetapkan utilisasi mesin (U) pada nilai 0.9 (90%). Berdasarkan data tersebut, waktu kedatangan *job* baru dapat diprediksi dengan menggunakan persamaan 3 untuk memenuhi kondisi distribusi eksponensial.

$$nextTime = \frac{-\ln(U)}{\lambda} \qquad (3)$$

Untuk memodelkan kondisi stokastik pada kasus waktu proses yang tidak pasti kami menggunakan probabilitas *triangular distribution*. Sedangkan untuk kasus pilihan *job* mana yang akan dipilih sebagai ketika ada *job* baru yang datang, kami menggunakan probabilitas *uniform distribution*. Dikarenakan kondisi yang stokastik, agar hasil komparasi algoritma tetap objektif, kami menerapkan fungsi *random* pada kode implementasi yang dapat ditentukan *seed state* nya. Dengan fitur yang memungkinkan kondisi *seed state* untuk ditentukan kita bisa mendapatkan hasil keluaran *random* yang dapat di-*reproduce* kembali. Hal ini memungkinkan perbandingan yang lebih objektif.

4. Hasil dan Pembahasan

Hasil implementasi tahap awal dari aplikasi dapat dilihat pada Gambar 11. Tampilan pada sebelah kiri Gambar 11 merupakan deskripsi dari *job* yang akan dikalkulasi untuk proses penjadwalan. Sedangkan tampilan pada sebelah kanan Gambar 11 merupakan representasi penjadwalan berdasarkan data *initial solution*. Tampilan gantt-chart pada Gambar 11 yang ada di sebelah kanan, tidak merepresentasikan kondisi gantt-chart penjadwalan yang sebenarnya karena pada tahap ini gantt-chart yang ditampilkan menggunakan nilai a^2 yang merupakan waktu proses yang memiliki fungsi membership 1 pada TFN. Pada proses untuk mengkalkulasi hasil penjadwalan, aplikasi mempertimbangkan input berupa waktu proses dalam representasi TFN yang mana ini bukanlah merupakan data waktu proses yang sebenarnya.



Gambar 11 Tampilan aplikasi untuk memulai simulasi

Gambar 12 Tampilan aplikasi ketika simulasi mulai berjalan

Gantt-chart pada Gambar 12 sudah menggambarkan hasil penjadwalan dengan menggunakan data yang berasal dari distribusi waktu tertentu. Untuk setiap operasi pada Gambar 12, waktu prosesnya sudah menggunakan data keluaran dari fungsi probabilitas triangular distribution, dimana keluaran dari fungsi ini nilainya sudah berupa nilai tertentu yang spesifik. Keluaran nilai tertentu dari fungsi tersebut pada simulasi merepresentasikan durasi waktu proses yang terjadi pada kondisi sebenarnya. Untuk membandingkan hasil *mean flow time* antara aplikasi yang menggunakan fuzzy number dengan yang tidak menggunakannya, kami menjalankan fitur simulasi hingga 2000 *job* dirilis dan selesai diproses. Dataset yang digunakan untuk simulasi kali ini adalah *problem* abz6. Simulasi dimulai pada waktu 0, dimana kami mengkondisikan bahwa sudah ada 10 *job* yang tersedia dan siap untuk diproses sejak waktu 0.

Job-job berikutnya akan secara berkesinambungan dirilis pada waktu kedatangan berdasarkan persamaan 3. Gambar 13 dan Gambar 14 merupakan gambar yang menunjukan gantt-chart hasil penjadwalan dan *mean flow time* ketika *job* 37 dirilis masing-masing untuk yang menggunakan fuzzy number dan yang tidak menggunakan fuzzy number. Bila dibandngkan mean flow time dari dua gambar tersebut, maka nilai *mean flow time* yang lebih kecil adalah Gambar 13, yaitu yang menggunakan fuzzy number. Nilai *mean flow time* yang lebih optimal pada satu waktu saja tidak bisa menjamin bahwa metode yang diterapkan lebih baik, kita harus membandingkan juga nilai-nilai nya pada kondisi waktu yang lain.

Gambar 13 Mean flow time dengan fuzzy number ketika job 37

Gambar 14 Mean flow time tanpa fuzzy number ketika job 37

Jika nilai ini dianalisa dan dibandingkan terus hingga 2000 *job* selesai diproses akan didapatkan hasil yang cukup menarik seperti yang terlihat pada Tabel I.

Tabel 1.
Meanflow Time antara VNS + Fuzzy Number dan VNS – Fuzzy Number

Job ke	VNS + Fuzzy Number	VNS - Fuzzy Number
37	831.52	973.07
80	834.24	942.16
250	946.90	948.45
274	943.11	952.81
1229	997.86	1009.21
1505	1013.87	1018.97
2000	1071.81	1104.92

5. Kesimpulan

Hasil percobaan menunjukan bahwa setelah fuzzy number dikombinasikan dengan algoritma variable neighborhood search, mean flow time yang didapatkan dari hasil simulasi hampir selalu lebih baik jika dibandingkan dengan yang tidak menggunakan fuzzy number. Hal ini menunjukkan representasi fuzzy number efektif digunakan untuk meminimasi flow time pada kasus DJSS dengan kondisi fuzzy processing time. Aplikasi tahap awal yang diimplementasikan pada paper ini masih memiliki banyak kekurangan terutama dikarenakan fitur simulasinya belum diintegrasikan dengan teknik process mining untuk menghasilkan hasil simulasi yang lebih menggambarkan kondisi real berdasarkan data event log.

Referensi

[1] Y. N.Sotskov, N. V. Shakhlevich, NP-hardness of shop-scheduling problems with three jobs, *Journal of Discrete Applied Mathematics*, Volume 59, pp. 237 – 266, 1995.

[2] S. C. Lin, E. D. Goodmand, W. F. Punch. A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems, *1997 ICGA 7th International Conference on Genetic Algorithms*, East Lansing, MI, USA, Juli, pp. 488 – 488, 1997.

[3] M. Sakawa, R. Kubota., Two-Objective Fuzzy Job Shop Scheduling through Genetic Algorithm, *Journal of Electronic and Communication in Japan*, Part 3, Volume 84, No 4, pp. 60 – 68, 2001

[4] Q. Niu, B. Jiao, X. Gu, Particle Swarm Optimization Combined with Genetic Operators for Job Shop Scheduling Problem with Fuzzy Processing Time, *Journal of Applied Mathematics and Computation*, Volume 205, pp. 148 – 158, 2008.

[5] K. R. Baker, D. Trietsch, *Principles of Sequence and Scheduling*, John Wiley & Sons, 2009.

[6] A. Rozinat, R. Mans, M Song, and W. V. D. Aalst, Discovering simulation models, Information Systems, Volume 34, pp. 305 – 327, 2009

[7] M. A. Adibi, M. Zandieh, Dynamic Job Shop Scheduling using Variable Neighborhood Search, *International Journal of Production Research*, Volume 48, pp. 2449 – 2458, 2010.

[8] M. Wynn, A. Rozinat, W. V. D. Aalst, A. T. Hofstede, and C. Fidge, Process Mining and Simulation. *Modern Business Process Automation*, pp.437-457, 2009.

[9] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, Lulu, 2011.

[10] B. Liu, Y. Fan, Y. Liu, A Fast Estimation of Distribution Algorithm for a Dynamic Fuzzy Flexible Job-Shop Scheduling

Problem, *Journal of Computers & Industrial Engineering*, Volume 87, pp. 193 – 201, 2015.

[11] W. V. D. Aalst, *Process Mining: Data Science in Action*, Springer, 2016.

[12] P. Sharma, A. Jain, Stochastic Dynamic Job Shop Scheduling with Sequence-Dependent Setup Times Simulation Experiment, *International Journal of Engineering and Technology*, Volume 5, pp. 19 – 25, 2015.

[13] R. Stephens, *Beginning Software Engineering*, Wrox, 2015.

[14] R. Agustiansyah. Perancangan Aplikasi Pemantauan, Pengawasan serta Pengelolaan Rumah Sakit untuk Dinas Kesehatan Kota Bandung., *Jurnal Rekayasa Sistem dan Industri*, Volume 4, pp. 152 – 160, 2017.

[15] O. B. Hadad, M. Solgi, H. A. Loáiciga, *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*, John Wiley & Sons, 2017.

[16] J. Sharabi, M. A. Adibi, M. Mahootchi, A Reinforcement Learning Approach to Parameter Estimation in Dynamic Job Shop Scheduling, *Journal of Computers & Industrial Engineering*, Volume 110, pp. 75 – 82, 2017.

[17] T. Jamrus, C. F. Chien, M. Gen, K. Sethanan, Hybrid Particle Swarm Optimization Combined with Genetic Operators for Flexible Job-Shop Scheduling Under Uncertain Processing Time for Semiconductor Manufacturing, *Journal of IEEE Transactions on Semiconductor Manufacturing*, Volume 31, pp. 32 – 40, 2018.