BAB I
PENDAHULUAN

1.1 Latar Belakang

Scheduling (Penjadwalan) merupakan salah satu solusi untuk permasalahan yang sering terjadi dalam proses produksi saat ini. Penjadwalan adalah proses dalam pengambilan keputusan yang sangat krusial untuk menentukan sistem produksi dan manufaktur. Penjadwalan pada umumnya digunakan di berbagai bidang pekerjaan mulai dari kesehatan, industri, perbankan, jasa, perdagangan, dan sebagainya **(Pinedo, 2002)**. Penjadwalan adalah salah satu solusi yang penting di dalam bidang produksi dan berkaitan untuk menemukan urutan pekerjaan pada mesin tertentu untuk meminimalkan waktu penyelesaian semua pekerjaan **(Li, 2017)**.

Job Shop Scheduling Problem (JSSP) merupakan salah satu proses penjadwalan yang mengatur proses dalam mengatur sumber daya untuk menyelesaikan permasalahan yang meliputi pekerjaan, mesin, dan durasi. Tujuan dari permasalahan JSSP adalah untuk meminimalkan waktu penyelesaian dalam pekerjaan (*makespan*). *Makespan* adalah waktu yang diperlukan untuk menyelesaikan seluruh pekerjaan pada mesin yang ada. Permasalahan JSSP telah diselesaikan oleh beberapa algoritma seperti *Ant Colony Optimization* (ACO) **(Masruroh, 2009)**, *Simulated Anneling* (SA) **(Van Laarhoven, dkk, 1990)**, *Genetic Algorithm* (GA) **(Li dan Chen, 2010)**, dan *Teaching Learning Based Optimization* (TLBO) **(Reddy, 2016)**.

Teaching Learning Based Optimization (TLBO) adalah algoritma yang diperkenalkan oleh **Rao (2011)**. Algoritma ini terinspirasi dari proses belajar mengajar dalam suatu kelas. Pada TLBO terdapat dua fase yaitu fase mengajar dan fase pelajar. Pada fase mengajar, solusi terbaik pada fase ini disebut sebagai pengajar atau pelajar terbaik dan akan melatih pelajar lainnya untuk mendapatkan solusi terbaik. Sedangkan pada fase pelajar, pelajar akan belajar dari pengajar. Selain itu juga pelajar dapat belajar dari pelajar lainnya. Pada proses TLBO, semua pelajar belajar dari satu pengajar dan sesama pelajar yang berada di lingkungan tersebut **(Aidil, 2014)**. Kelebihan dari algoritma TLBO untuk menyelesaikan

permasalahan JSSP adalah tidak adanya parameter khusus (parameter seperti pada algoritma genetika) yang dibutuhkan dalam proses penyelesaian, sehingga dalam menentukan solusi akan lebih mudah **(Weng, 2017)**.

Tabu Search (TS) adalah algoritma yang diperkenalkan oleh **Glover (1986)**. Algoritma TS merupakan metode optimasi yang berbasis pada *local search*. Ide dasar dari algoritma TS adalah mencegah proses pencarian dari *local search* agar tidak melakukan proses pencarian ulang pada ruang solusi yang pernah ditelusuri, dengan memanfaatkan suatu struktur memori yang mencatat sebagian jejak proses pencarian yang telah dilakukan **(Amico dan Trubian, 1993)**. Kelebihan dari algoritma TS adalah berfungsi untuk menghindari diterimanya kembali solusi sebelumnya menjadi *current solution* pada iterasi selanjutnya **(Fred dkk, 1995)**.

Alasan dilakukannya *hybrid* antara algoritma TLBO dan TS karena TLBO merupakan algoritma yang berdasarkan populasi yang cocok dalam melakukan kemampuan eksplorasi untuk menemukan solusi, namun buruk dalam melakukan eksplorasi solusi secara lokal **(Kumar, 2018)**. Oleh karena itu, untuk meningkatkan kemampuan dalam memanfaatkan ruang solusi secara lokal, dilakukan kombinasi dengan algoritma TS yang memiliki kemampuan *local search* yang baik. Berdasarkan uraian yang telah dijelaskan, maka sangat menarik untuk membahas penyelesaian *Job Shop Scheduling Problem* (JSSP) dengan menerapkan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka diperoleh rumusan masalahnya sebagai berikut :

1. Bagaimana menyelesaikan *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS)?
2. Bagaimana membuat program penyelesaian *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS)?

3. Bagaimana implementasi program penyelesaian *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS) pada contoh kasus?

1.3 Tujuan

Berdasarkan latar belakang yang telah dijelaskan, maka diperoleh tujuan sebagai berikut :

1. Menyelesaikan *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS).
2. Membuat program penyelesaian *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS).
3. Implementasi program penyelesaian *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS).

1.4 Manfaat

Berdasarkan latar belakang yang telah dijelaskan, maka diperoleh manfaat sebagai berikut:

1. Menambah wawasan untuk menyelesaikan permasalahan *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS)
2. Diharapkan dapat menjadi referensi alternatif untuk penerapan algoritma lainnya untuk menyelesaikan permasalahan *Job Shop Scheduling Problem* (JSSP).
3. Implementasi program yang telah dibuat diharapkan dapat membantu dalam permasalahan produksi.

BAB II
TINJAUAN PUSTAKA

Dalam penulisan penelitian ini, diperlukan beberapa informasi dan definisi mengenai yang digunakan untuk memperdalam dan mempermudah dalam menyelesaikan permasalahan *Job Shop Scheduling Problem* (JSSP) dengan menggunakan *hybrid* algoritma *Teaching Learning Based Optimization* (TLBO) dan *Tabu Search* (TS).

2.1 *Scheduling* (Penjadwalan)

Penjadwalan (*scheduling*) merupakan salah satu unsur yang sangat penting dalam memulai suatu pekerjaan atau produksi. Menurut **Pinedo (2002)** penjadwalan merupakan masalah yang sangat penting dalam melakukan perencanaan dan manajemen dalam suatu perusahaan. Untuk menentukan penjadwalan terbaik bisa sangat mudah, namun bisa juga sangat sulit karena berdasarkan beberapa parameter.

Penjadwalan biasanya dibutuhkan dalam pengaturan operasional, mesin, sumber daya manusia, dan sebagainya. Oleh karena itu, menurut **Bedworth (1987)** tujuan dari penjadwalan ada 4, yaitu:

1. Dapat meningkatkan penggunaan sumber daya yang dimiliki.
2. Dapat mengurangi sejumlah pekerjaan yang mengganggu dalam antrian ketika sumber daya yang dimiliki sedang sibuk.
3. Dapat mengurangi terjadinya keterlambatan dalam setiap pekerjaan yang dapat memunculkan *penalty cost* (biaya keterlambatan).
4. Dapat membantu dalam menentukan keputusan saat sedang melakukan perencanaan.

2.1.1 Elemen Penjadwalan

Menurut **Baker (2009)** pada penjadwalan terdapat 3 elemen yaitu:

1. Pekerjaan

Pekerjaan merupakan pekerjaan yang harus diselesaikan dalam suatu waktu. Biasanya dalam permasalahan diberikan lebih dari satu pekerjaan untuk dijadwalkan.

2. Operasi

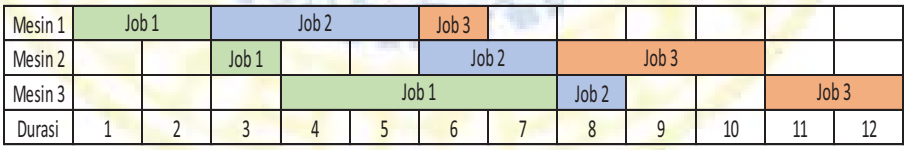
   Operasi merupakan proses yang dibutuhkan dalam menyelesaikan suatu pekerjaan. Operasi bertujuan untuk memberikan urutan dalam proses pengerjaan pekerjaan yang telah dijadwalkan. Operasi biasanya disajikan dalam tabel operasi.

3. Mesin

   Mesin merupakan sumber daya yang diperlukan dalam menyelesaikan suatu pekerjaan. Satu mesin hanya bisa melakukan satu pekerjaan dalam kondisi tertentu.

2.1.2 *Gantt Chart*

Gantt chart merupakan diagram perencanaan yang digunakan untuk menyelesaikan penjadwalan sumber daya dan alokasi waktu. Tujuan dari *gantt chart* adalah untuk memastikan bahwa semua kegiatan telah sesuai dengan yang direncanakan, urutan kerja telah diperhitungkan, perkiraan waktu telah tercatat, dan keseluruhan waktu proyek telah dibuat **(Widyastuti dkk ,2019)**. Salah satu contoh untuk permasalahan penjadwalan 3 mesin dan 3 pekerjaan dapat ditunjukkan pada **Gambar 2.1**.



Gambar 2. 1 Contoh Permasalahan Penjadwalan 3 mesin dan 3 pekerjaan

Pada contoh *gantt chart* pada **Gambar 2.1**, terdapat 3 pekerjaan (pekerjaan 1, pekerjaan 2, dan pekerjaan 3) yang berada pada sumbu horizontal dan 3 mesin (Mesin 1, Mesin 2, dan Mesin 3) yang berada pada sumbu vertikal. Pekerjaan 1 dijadwalkan pada awal Mesin 1, kemudian pekerjaan *2* dijadwalkan pada Mesin 1 setelah pekerjaan 1 selesai dijadwalkan. Penjadwalan pekerjaan 1 dilakukan di Mesin 3 setelah penjadwalan pekerjaan 1 di Mesin 2 selesai, begitu juga dengan

pekerjaan 2 di Mesin 3, dijadwalkan setelah pekerjaan 2 di Mesin 2 selesai dikerjakan. Dan seterusnya sampai pekerjaan 3 di Mesin 3. Sehingga, diperoleh nilai *makespan* sebesar 12 satuan waktu.

2.2 *Job Shop Scheduling Problem* (JSSP)

Tujuan dari *Job Shop Scheduling Problem* (JSSP) adalah untuk meminimalkan nilai *makespan* total waktu yang telah di jadwalkan untuk menyelesaikan pekerjaan. Menurut **Puspitasari (2016)**, terdapat tiga kelebihan pada JSSP. Pertama, dengan adanya penjadwalan secara efektif maka mesin dapat digunakan secara efektif dan memperoleh keuntungan yang lebih besar. Kedua, penjadwalan dapat menambah kapasitas dan fleksibilitas yang terkait, serta memberikan waktu pengiriman yang lebih cepat. Ketiga, keuntungan dari penjadwalan yang baik adalah keunggulan kompetitif dalam pengiriman.

Menurut **Gen dan Cheng (1997)** permasalahan dari JSSP dapat dituliskan dalam bentuk matematis seperti berikut:

Fungsi tujuan:

$$\min \max_{1\le k\le m} \{\max_{1\le i\le n}\{c_{ik}\}\} \quad (2.1)$$

Dengan kendala:

$$c_{ik} - t_{ik} + M(1 - a_{ihk}) \ge c_{ik} \quad (2.2)$$

$i = 1, 2, \dots, n$ dan $h, k = 1, 2, \dots, m$

$$c_{jk} - c_{ik} + M\left(1 - x_{ijk}\right) \ge t_{jk} \quad (2.3)$$

$i, j = 1, 2, \dots, n$ dan $h, k = 1, 2, \dots, m$

$$c_{ik} \ge 0, i = 1, 2, \dots, n \text{ dan } k = 1, 2, \dots, m \quad (2.4)$$

$$a_{ihk} = \{0, 1\}, i = 1, 2, \dots, n \text{ dan } h, k = 1, 2, \dots, m \quad (2.5)$$

$$x_{ijk} = \{0, 1\}, i, j = 1, 2, \dots, n \text{ dan } k = 1, 2, \dots, m \quad (2.6)$$

Dengan keterangan:

c_{ik} : waktu selesai pengerjaan pekerjaan-*i* pada mesin-*k*.

c_{jk} : waktu selesai pengerjaan pekerjaan-*j* pada mesin-*k*.

t_{ik} : durasi pengerjaan pekerjaan pekerjaan-*i* pada mesin-*k*.

t_{jk} : durasi pengerjaan pekerjaan pekerjaan-*i* pada mesin-*k*.

M : bilangan bulat besar positif.

a_{ihk} : koefisien indikator.

x_{ijk} : variabel indikator.

Dengan,

$$a_{ihk} = \begin{cases} 1, & \text{jika mesin} - h \text{ mendahului mesin} - k \text{ pada pekerjaan} - i \\ 0, & \text{lainnya} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{jika pekerjaan} - i \text{ mendahului pekerjaan} - j \text{ pada mesin} - k \\ 0, & \text{lainnya} \end{cases}$$

Persamaan (2.1) merupakan fungsi tujuan yaitu untuk meminimalkan nilai *makespan*. Persamaan (2.2) sampai dengan (2.4) merupakan batasan-batasan yang harus dipenuhi. Batasan (2.2) memastikan urutan pengerjaan operasi setiap pekerjaan sesuai dengan urutan yang ditentukan. Batasan (2.3) memastikan bahwa setiap mesin dapat memproses hanya satu pekerjaan pada satu waktu, dan batasan (2.4) memastikan bahwa waktu selesai pengerjaan pekerjaan-*i* pada mesin-*k*.harus lebih besar sama dengan 0.

2.2.1 Metode *Giffler* dan *Thompson*

Metode *Giffler* dan *Thompson* merupakan metode yang digunakan dalam menyelesaikan *job shop scheduling problem*. Metode *Giffler* dan *Thompson* merupakan metode yang digunakan untuk menyelesaikan permasalahan JSSP dengan meminimalkan waktu luang yang ada penjadwalan dengan cara menggeser operasi untuk mendapatkan nilai *makespan* lebih kecil yang disebut dengan semi-aktif dan suatu penjadwalan dengan *delay time* yang tidak memungkinkan menggeser operasi lagi disebut aktif **(Kobayashi dkk, 1995)**. Inti dari metode *Giffler* dan *Thompson* adalah metode yang digunakan untuk mengaktifkan solusi dari seluruh pekerjaan yang ada pada JSSP.

2.3 *Teaching Learning Based Optimization* (TLBO)

Teaching Learning Based Optimization (TLBO) merupakan algoritma *meta heuristic* yang diperkenalkan oleh **Rao (2011)**. Algoritma ini terinspirasi dari proses belajar mengajar pada kehidupan manusia. Fase belajar mengajar yang dimaksud adalah seperti guru yang mengajar pada muridnya di suatu kelas. Pada TLBO terdapat 2 fase yaitu fase pengajar dan fase pelajar.

2.3.1 Fase Pengajar

Pengajar memiliki ilmu yang lebih tinggi daripada pelajar. Oleh karena itu, pada fase ini pelajar belajar dari pengajar. Selain itu juga, pengajar berusaha meningkatkan nilai dari para pelajar untuk mendapatkan nilai terbaik mereka. Nilai yang terdapat pada solusi direpresentasikan sebagai $X_{j,k,i}$. Untuk setiap iterasi *i*, diasumsikan terdapat mata pelajar *m* untuk setiap mata pelajaran *j* dan pembelajar *k*. Solusi dari pelajar atau yang biasa disebut dengan nilai *fitness* diperhitungkan berdasarkan pada semua mata pelajaran yang didapatkan oleh populasi pelajar yang dianggap sebagai hasil dari pembelajar.

Berdasarkan **Rao (2011)** rumus yang digunakan pada fase pengajar adalah sebagai berikut :

$$T_f = round[1 + rand(0,1)] \quad (2.7)$$

$$DifferenceMean_{j,k,i} = r_i(X_{j,k\ best,i} - T_f M_{j,i}) \quad (2.8)$$

$$X_{j,k,i}^{new} = X_{j,k,i}^{old} + Diffenernce Mean_{\ j,k,i} \quad (2.9)$$

$$X_{j,k,i}^{new} = X_{j,k,i}^{old} + r_i(X_{j,k\ best,i} - T_f M_{\ j,i}) \quad (2.10)$$

Dengan keterangan :

T_f : faktor ajaran yang menentukan nilai rata-rata harus diubah.

$rand(0,1)$: bilangan real yang dibangkitkan secara acak pada interval (0, 1).

$DifferenceMean_{j,k,i}$: perbedaan antara hasil rata-rata pada mata pelajaran.

r_i : bilangan real yang dibangkitkan secara acak pada interval (0, 1) pada iterasi ke-*i.*

$M_{j,i}$: rata-rata nilai pada mata pelajaran ke-*j* pada iterasi ke-*i,*

$x_{j,k,i}$: nilai pada mata pelajaran ke-*j* pada pelajar ke-*k* pada iterasi ke-*i.*

$x_{j,k,best\ i}$: hasil pembelajaran terbaik *k* pada mata pelajaran ke-*j* pada iterasi ke-*i.*

Persamaan (2.9) dan (2.10) digunakan untuk memperbarui solusi dalam fase pengajar dengan $X_{new\ j,k,i}$ merupakan nilai yang telah diperbarui dari $X_{old\ j,k,i}$. $X_{new\ j,k,i}$ akan diterima jika menghasilkan nilai *fitness* yang lebih baik. Semua nilai *fitness* yang diterima pada proses akhir di fase pengajar akan dipertahankan dan

nilai-nilai tersebut akan menjadi nilai acuan untuk fase pelajar. Maka dari itu, fase pelajar bergantung kepada fase pengajar.

2.3.2 Fase Pelajar

Pada fase ini, pelajar belajar dari pengajarnya. Namun selain belajar dari pengajar, pelajar juga dapat meningkatkan pengetahuan mereka dengan cara berinteraksi dengan pelajar yang lainnya. Seorang pelajar belajar hal baru dari pelajar lainnya yang memiliki pengetahuan lebih. Interaksi yang dilakukan oleh para pelajar diasumsikan sebagai dua pelajar secara acak yang saling berinteraksi yaitu pelajar P dan pelajar Q dengan $fitness_{P,i} \neq fitness_{Q,i}$.

Berdasarkan **Rao (2011)** rumus yang digunakan pada fase pelajar adalah sebagai berikut :

Untuk fungsi tujuan minimum, digunakan persamaan :

$$X_{j,P,i}^{new} = X_{j,P,i}^{old} + r_i(X_{j,P,i}^{old} - X_{j,Q,i}^{old}) \quad (2.11)$$

$$X_{j,P,i}^{new} = X_{j,P,i}^{old} + r_i(X_{j,Q,i}^{old} - X_{j,P,i}^{old}) \quad (2.12)$$

Untuk fungsi tujuan maksimum, digunakan persamaan :

$$X_{j,P,i}^{new} = X_{j,P,i}^{old} + r_i(X_{j,Q,i}^{old} - X_{j,P,i}^{old}) \quad (2.13)$$

$$X_{j,P,i}^{new} = X_{j,P,i}^{old} + r_i(X_{j,P,i}^{old} - X_{j,Q,i}^{old}) \quad (2.14)$$

Dengan penjelasan :

Untuk permasalahan dengan fungsi tujuan minimum, jika nilai $fitness_{P,i}$ lebih baik dibandingkan nilai $fitness_{Q,i}$ maka gunakan persamaan (2.11) sedangkan jika dengan fungsi tujuan maksimum, maka gunakan persamaan (2.13). Sebaliknya, jika nilai $fitness_{P,i}$ lebih buruk dibandingkan nilai $fitness_{Q,i}$ maka gunakan persamaan (2.12) untuk permasalahan dengan fungsi tujuan minimum dan persamaan (2.14) untuk permasalahan dengan fungsi tujuan maksimum dan nilainya akan diterima jika nilai *fitness* lebih baik.

2.4 *Tabu Search* (TS)

Algoritma *Tabu Search* (TS) diperkenalkan oleh **Fred Glover (1997)**. Algoritma TS merupakan metode optimasi yang berbasis pada *local search*. Algoritma TS merupakan metode optimasi matematis yang termasuk ke dalam

kelas *local search* **(Suyanto, 2010)**. Algoritma TS memperbaiki penampilan *local search* dengan memanfaatkan penggunaan struktur *memory*. Sebagian solusi yang pernah dibangkitkan ditandai sebagai "*tabu"* (yang berarti sesuatu yang terlarang), sehingga algoritma TS tidak akan mendatangi solusi tersebut secara berulang kali.

Menurut **Rera dan Budi (2010)** berikut ini merupakan kerangka umum dari algoritma TS :

1. Tahap Inisialisasi

   Pada tahap ini, dilakukan penentuan ukuran *tabu list* awal.
2. Membangkitkan Kandidat Solusi

   Pada tahap ini, dilakukan membangkitkan kandidat solusi dari solusi tetangga (*neighbour solution*).
3. Pemilihan Solusi Terbaik Pada Kandidat Solusi

   Pada tahap ini, kandidat solusi yang ada dihitung *T(i)* sebagai fungsi tujuannya. Kemudian dipilih solusi terbaik pada kandidat solusi.
4. *Update Tabu List*

   Pada tahap ini, solusi terbaik yang dipilih pada langkah 3 akan dimasukan ke dalam *tabu list*.
5. Ulangi Langkah 2 sampai dengan Langkah 4

   Pengulangan dilakukan hingga mencapai iterasi maksimum yang ditentukan. Proses berhenti menunjukkan solusi terbaik yang dihasilkan adalah solusi yang didapat dengan menggunakan algoritma ini.

2.5 Bahasa Pemrograman Java

Java merupakan bahasa pemrograman yang didasari oleh OOP (Oriented Object Programming) yaitu merupakan teknik membuat suatu program berdasarkan objek. Menurut **Huda (2010)**, hal yang paling penting dalam pemrograman Java adalah memahami karakter dari pola pemrograman berbasis objek yang mencakup konsep utama pada *Object Oriented Programming* (OOP) yaitu:

1. *Class*

   Dalam Java, kelas didefinisikan menggunakan kata kunci *class*.
2. *Method*

Terdapat dua buah *method* (metode) yaitu fungsi dan prosedur. Fungsi merupakan metode yang memiliki nilai balik dengan menggunakan kata kunci *tipe_data <spasi> nama_fungsi()*. Sebaliknya prosedur merupakan metode yang tidak memiliki nilai balik yang menggunakan kata kunci *void <spasi> nama_fungsi()*.

3. *Inheritance*

   Inheritance (Pewarisan) adalah membentuk subkelas baru (kelas anak) dari kelas utama atau kelas induk sebelumnya yang menggunakan kata kunci *class <spasi> nama_kelas_anak <spasi> extends <spasi> nama_kelas_induk*.

4. *Polimorfisme*

   Polimorfisme adalah pembentukan kelas baru yang bersifat abstrak karena adanya keragaman fungsi dari objek-objek yang identik. Oleh karena itu *Polimorfisme* membentuk kelas abstrak yang menggunakan kata kunci *abstract*.

5. *Interface* hampir menyerupai kelas abstrak, akan tetapi *interface* merupakan kelas abstrak sepenuhnya yang bertujuan untuk menerapkan pewarisan jamak. *Interface* menggunakan kata kunci *Interface*.