Penerapan Constraint Satisfaction Problem pada Permasalahan Job Shop Scheduling dengan Menggunakan Algoritma Backtracking

Sabrina Diza Melinda[1], Zahra Aulia Firdausi[2], Syukriyatul Hanifa[3], Bima Aulia[4], Hendrika Anggriawan[5]
Departemen Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, IPB University Jalan Raya Dramaga, Kampus IPB Dramaga, Bogor, 16680, Jawa Barat, Indonesia
Email:[1]sabrina_dizam21@apps.ipb.ac.id, [2]zahra_firdausi@apps.ipb.ac.id, [3]syukriyatul_hanifa@apps.ipb.ac.id, [4]bima_aulia@apps.ipb.ac.id, [5]hendrika_ang@apps.ipb.ac.id

Abstrak:

Penjadwalan dalam suatu proses produksi merupakan salah satu hal yang paling penting dalam bidang industri. Optimal atau tidaknya suatu penjadwalan sangat mempengaruhi waktu penyelesaian pekerjaan. Untuk menghasilkan penjadwalan yang optimal, jadwal penggunaan mesin dan pekerja harus diatur sedemikian rupa sehingga tidak terjadi tabrakan antara penggunaan mesin yang satu dengan yang lainnya. Penelitian ini bertujuan untuk mengatasi masalah *job shop* tersebut dengan menggunakan algoritma *backtracking.* Langkah awal yang dilakukan adalah dengan mempresentasikan *job shop scheduling* tersebut sebagai sebuah *constraint satisfaction problem (CSP)*, kemudian dilakukan proses pencarian solusi CSP tersebut menggunakan algoritma *backtracking*. Hasil dari penelitian ini adalah suatu slot jadwal proses produksi yang memenuhi *constraint* dan waktu yang paling optimal untuk menyelesaikan suatu proses produksi tersebut.

Kata kunci : algoritma backtracking, constraint satisfaction problem (CSP), job shop scheduling

Abstract:

Scheduling in a production process is one of the most important things in industry. Time to complete a job is affected by whether a schedule is optimal or not. To make an optimal schedule, the schedule for machines and workers must be arranged in such a way to avoid an overlap. The purpose of this research is to solve this job shop problem using a backtracking algorithm. The first step to solve this problem is present the job shop scheduling as a constraint satisfaction problem (CSP), then use the backtracking algorithm to carry out the process to find a solution. The result of this research is a slot schedule of the production process that qualifies the constraint and the most optimal time to finish the production process.

Keywords : backtracking algorithm, constraint satisfaction problem (CSP), job shop scheduling

I. PENDAHULUAN

Pesatnya kemajuan di bidang teknologi informasi membuat banyak industri yang ingin memotong rantai pasok dari produsen ke konsumen berlomba-lomba untuk menerapkan teknologi yang terbaik demi meningkatkan bisnisnya. Banyak industri melakukan ini dengan harapan dapat menjual produknya dengan keuntungan yang lebih besar dan konsumen tetap dapat membeli produk mereka dengan harga yang relatif lebih murah. Hal ini juga dapat membuat hubungan produsen dan konsumen menjadi lebih dekat.

Kedekatan antara produsen dan konsumen yang terbentuk membuat konsumen bisa mengungkapkan secara langsung pada produsen seperti apa produk yang mereka mau. Dalam menjaga hubungan ini, salah satu cara yang dapat dilakukan oleh seorang produsen

adalah dengan membuat produk custom untuk setiap konsumennya.

Pembuatan produk custom memaksa produsen/industri untuk menyerahkan seluruh proses pembuatan satu produk custom pada satu pekerja saja. Produk custom biasanya membutuhkan proses yang unik yang harus dikerjakan secara berurutan. Oleh karena itu, pekerja tersebut harus menggunakan mesin jenis yang berbeda-beda, namun tetap sesuai dengan urutan.

Untuk dapat membuat banyak produk custom, industri harus menyusun strategi agar seluruh pekerjaan dapat diselesaikan dengan cepat. Seringkali mesin yang dimiliki industri jumlahnya terbatas. Agar pekerjaan bisa diselesaikan dengan cepat, perlu dilakukan penjadwalan penggunaan mesin. Dengan adanya penjadwalan ini, diharapkan pekerjaan dapat diselesaikan secara cepat dan efisien.

II. Landasan Teori

A. Penjadwalan *Job-Shop*

Penjadwalan merupakan pengaturan alokasi sumber daya untuk menyelesaikan tugas-tugas yang melibatkan pekerjaan, sumber daya dan waktu[1]. Penjadwalan merupakan suatu cara yang dilakukan oleh manusia untuk bisa memanfaatkan waktu dengan lebih efisien sehingga dapat mencapai suatu tujuan rutinitas yang maksimal[2]. Pekerjaan yang harus diselesaikan memiliki batas waktu yang mempengaruhi prioritas pekerjaan. Penjadwalan job shop merupakan proses pengurutan pekerjaan yang harus melewati beberapa mesin dan urutan proses yang ditempuh masing-masing pekerjaan atau berbeda[1]. Proses dalam menyusun semua operasi dari semua job pada tiap mesin dilakukan dengan ketelitian yang tinggi, karena terdapat batasan (*Constraint*) yang harus dipenuhi [3]. Sehingga keseluruhan job dapat diproses menurut urutan pengerjaannya merupakan objek dari masalah penjadwalan job shop[1][2].

Terdapat 3 batasan (*Constraint*) utama untuk Job Shop *Problem* :

a. Tidak ada *task* yang dapat dimulai untuk suatu pekerjaan sampai *task* sebelumnya untuk pekerjaan itu diselesaikan.
b. Sebuah mesin hanya bisa bekerja pada satu task di satu waktu yang sama.
c. Sebuah tugas apabila sudah dimulai, maka harus dikerjakan hingga selesai.

B. *Constraint Satisfaction Problem*

Constraint satisfaction problem atau CSP merupakan sebuah pendekatan untuk mencari solusi dari suatu permasalahan dengan cara mencari objek / keadaan yang memenuhi kriteria atau batasan dari permasalahan tersebut. CSP terdiri atas 3 komponen utama yaitu:

1. *Constraint*

   Constraint merupakan suatu aturan yang ditentukan, yang akan mengatur nilai yang boleh diisikan ke dalam variabel atau kombinasi variabel. Constraint terdiri atas 3 macam yaitu *unary* (menyatakan sebuah variabel), *binary* (menyatakan persyaratan sepasang variabel), *n-ary* (menyatakan persyaratan tiga atau lebih variabel), dan (menyatakan syarat yang sebaiknya dipenuhi, tetapi tidak harus terpenuhi)
2. Domain

   Domain merupakan nilai legal yang dapat diisi ke dalam variabel. Sebuah domain akan membatasi nilai suatu variabel.
3. Variabel

   Variabel merupakan suatu penampung yang dapat diisi dengan berbagai nilai[9].

C. *Algoritma Backtracking*

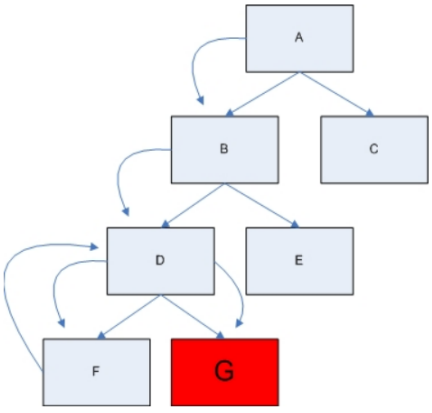
Algoritma Backtracking pertama kali diperkenalkan oleh D.H Lehmer pada tahun 1950. Algoritma ini merupakan salah satu metode pemecahan masalah yang termasuk dalam strategi yang berbasis pencarian pada ruang status, yang bekerja secara rekursif dan sistematis terhadap semua kemungkinan solusi yang ada. Prinsip dasar algoritma backtracking adalah mencoba semua kemungkinan solusi yang ada dimana semua solusi dibuat dalam bentuk pohon solusi (*tree*) yang kemudian ditelusuri secara DFS (*Depth First Search*) sehingga dapat ditemukan solusi terbaik yang diinginkan [4][6]. Algoritma backtracking memiliki 3 properti utama dalam penerapannya, yaitu [4]:

1. Solusi persoalan. Solusi ini dinyatakan sebagai *vector* dengan *n-tuple*. $X=(x_1, x_2, \ldots, x_n)$, x_i merupakan himpunan berhingga S_i.

2. Fungsi pembangkit, dinyatakan sebagai T(k). Fungsi ini berfungsi untuk membangkitkan nilai X_k yang merupakan vektor solusi.
3. Fungsi pembatas. Fungsi ini digunakan untuk menentukan apakah (x_1, x_2, …, x_k) mengarah ke solusi atau tidak.

Langkah-langkah pencarian solusi menggunakan algoritma backtracking yaitu [5]:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Simpul yang sudah dibentuk dinamakan simpul hidup dan simpul hidup yang sudha diperluas tersebut dinamakan simpul-E (*Expand- node*)
2. Saat lintasan yang dihasilkan dari perluasan simpul-E tidak menghasilkan solusi yang diinginkan maka simpul tersebut akan menjadi simpul mati yang mana simpul tersebut tidak akan diperluas lagi.
3. Saat posisi terakhir berada di simpul mati, maka pencarian selanjutnya dilakukan dengan membangkitkan simpul *child* yang lain dan jika tidak ada simpul *child* tersebut maka dilakukan *backtracking* ke simpul *parent*.
4. Saat solusi sudah ditemukan atau tidak ada lagi simpul hidup yang perlu di-*backtrack* maka pencarian dihentikan.



Gambar 1. Contoh Algoritma Backtracking

Pseudocode untuk algoritma backtracking adalah [6]:

```
procedure Backtrack (input k:integer)
{mencari semua solusi persoalan
dengan metode backtracking (rekursif)
input: k,yaitu indeks komponen vektor
solusi, x[k]
output: solusi x = (x[1], x[2], …,
x[n])}

Algoritma:

For untuk setiap x[k] yang belum
dicoba sedemikian sehingga

  (x[k] <- T(k)) and B(x[1],
  x[2],..., x[k])= true do

  if (x[1], x[2], ... ,x[k]) adalah
  path solusi
  then
    CetakSolusi(x)
  endif
    Backtrack(k+1) {tentukan nilai
    untuk x[k+1]}
endfor
```

III. METODOLOGI PENELITIAN

Perancangan sistem dalam penelitian ini dilakukan dalam empat tahap, yakni tahap pengumpulan data, tahap pendefinisian *constraint*, tahap penyusunan algoritma, dan tahap evaluasi.

1. Pengumpulan Data

Data yang dikumpulkan untuk penelitian ini bersumber dari *website* http://mistic.heig-vd.ch/. Data yang dipakai adalah jenis mesin, lama waktu penggunaan mesin, dan jumlah pekerja. Terdapat 15 pekerja yang akan menggunakan 15 mesin dengan durasi pemakaian yang berbeda.

Mesin yang ada pada data input diasumsikan bekerja secara *dependent*, artinya input suatu mesin bisa merupakan bahan baku ataupun berupa *output* dari mesin lainnya. Pada data *input* yang kami gunakan terlihat bahwa susunan urutan mesin yang digunakan oleh para pekerja seperti acak, namun sebenarnya tidak. Hal itu karena semua mesin yang digunakan diasumsikan merupakan *multi-purpose machines* (MPM). MPM adalah mesin yang dilengkapi dengan alat-alat yang berbeda. Mesin dapat memproses pekerjaan hanya jika mesin tersebut dilengkapi dengan alat yang dibutuhkan oleh pekerja[10]. Sehingga, data input yang kami gunakan kemungkinan merupakan data *dependent*.

2. Pendefinisian *constraint*

Aturan-aturan yang harus dipenuhi dalam menyusun penjadwalan ini antara lain :

- Suatu *task* tidak akan dimulai jika *task* sebelumnya dari *job* yang sama belum selesai dilakukan.
- Sebuah mesin hanya bisa melakukan satu *task* diwaktu yang sama.
- Jika suatu *task* sudah dijalankan maka *task* tersebut harus dijalankan hingga akhir.

3. Penyusunan algoritma

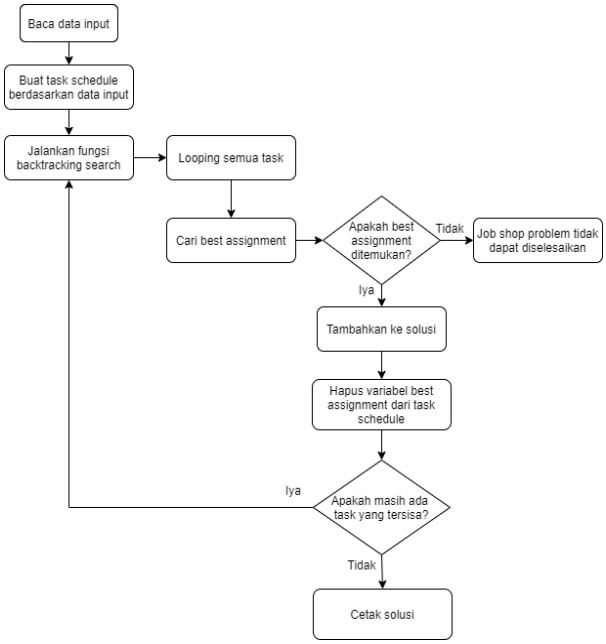
Algoritma yang digunakan dalam menyelesaikan permasalahan penjadwalan *job shop* ini adalah algoritma *backtracking*. Algoritma yang paling banyak dipakai untuk melakukan pencarian sistematik untuk menyelesaikan CSP adalah *backtracking*. Algoritma *backtracking* search (penelusuran kembali) adalah suatu bentuk algoritma *depth-first-search*. Backtracking dapat dilihat sebagaimana *searching* dalam *tree*, dimana setiap node mewakili state dan turunan dari setiap node mewakili ekstensi dari state tersebut.

4. Evaluasi

Tahap terakhir yang dilakukan adalah tahap evaluasi. Tahap ini dilakukan dengan membandingkan lama waktu penyelesaian secara *brute force* dengan penyelesaian dengan menggunakan algoritma *backtracking* untuk mengetahui kesesuaian penggunaan CSP dengan algoritma *backtracking* pada kasus penjadwalan *Job Shop* ini. Selain itu, hasil analisis ini akan dibandingkan dengan solusi optimal *parallel taboo search*.

IV. Hasil dan Diskusi

A. Perancangan Sistem



Gambar 2. Flowchart proses algoritma penjadwalan *job shop*

1. Baca data input

Data input terdiri dari banyak pekerja dan jumlah mesin. Untuk setiap pekerja didefinisikan urutan pekerjaan yang harus diselesaikan yang terdiri atas id mesin yang harus digunakan dan waktu pengerjaannya.

2. Buat *task schedule*

Dari data input yang telah didefinisikan kemudian dibuat task schedule menggunakan tipe data dictionary yang ada pada bahasa pemrograman python.

3. Jalankan fungsi *backtracking search*

Pencarian solusi dilakukan dengan menerapkan fungsi *backtracking search*. *Backtracking search* dijalankan secara rekursif untuk mencari task dengan waktu kinerja yang paling minimal dan tetap memenuhi constraint yang ada. Apabila tidak ditemukan task yang memenuhi constraint maka dapat disimpulkan bahwa permasalahan tidak dapat diselesaikan.

4. Cetak solusi

Setelah semua task selesai dijadwalkan maka solusi untuk job shop scheduling dapat dicetak. Bentuk

akhir dari solusi permasalahan ini adalah berupa jadwal penggunaan dari setiap mesin yang ada.

B. Uji Coba

Kriteria Penjadwalan

Pada proses uji coba terdapat kriteria penjadwalan sebagai berikut:

1. Input penjadwalan pekerja berupa alokasi mesin yang bekerja secara *dependent* dan durasi pemakaiannya.
2. Output penjadwalan berupa jadwal yang optimal. Jadwal penggunaan mesin dan pekerja yang dibutuhkan untuk menyelesaikan sebuah *task* akan diatur sedemikian rupa agar tidak terjadi tabrakan antara penggunaan mesin yang satu dengan yang lainnya.

Penjadwalan Produksi

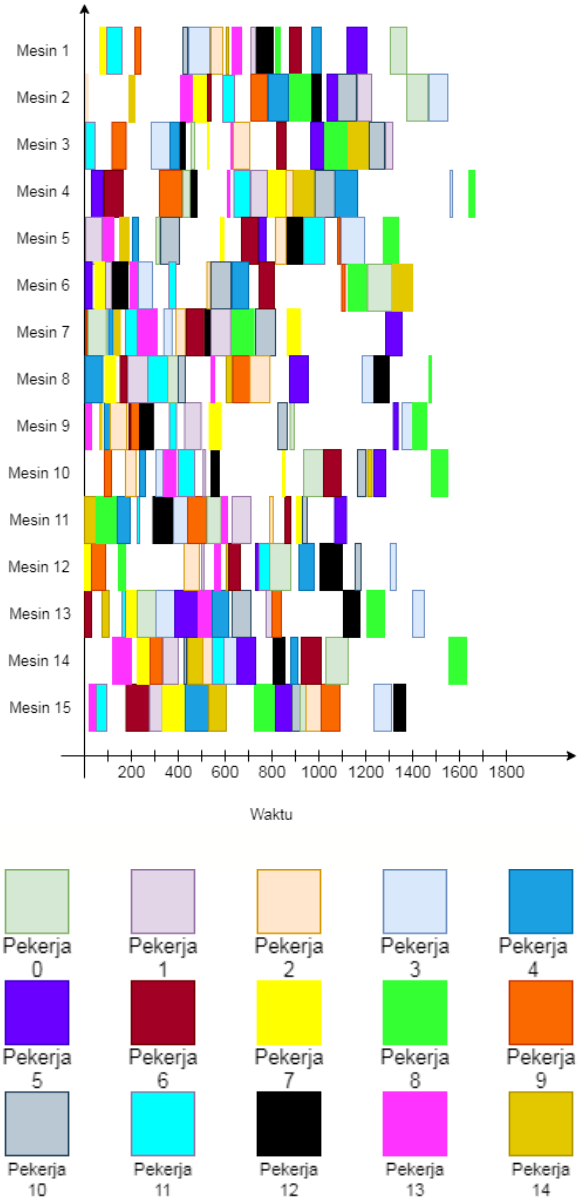
Penjadwalan produksi yang dilakukan menggunakan 15 jenis mesin dengan alokasi waktu yang berbeda. Kemudian Terdapat 15 pekerja yang harus menyelesaikan semua *task* dengan jumlah mesin yang sudah ditentukan.

Pada tahap ini, hal yang dilakukan adalah pengujian. Berikut adalah lama waktu (menit) penggunaan mesin setiap pekerja.

		Pekerja 0	Pekerja 1	Pekerja 2	Pekerja 3	Pekerja 4	Pekerja 5	Pekerja 6	Pekerja 7	Pekerja 8	Pekerja 9	Pekerja 10	Pekerja 11	Pekerja 12	Pekerja 13	Pekerja 14
Task 1	Mesin	7	5	2	6	8	6	13	12	11	7	5	3	6	9	11
	Waktu	94	74	4	73	78	29	18	32	85	5	90	47	65	28	57
Task 2	Mesin	13	6	9	3	9	4	4	6	12	12	8	15	9	15	9
	Waktu	66	31	82	23	23	61	75	52	30	59	27	43	62	21	16
Task 3	Mesin	5	8	10	10	7	13	8	1	7	10	14	1	11	5	13
	Waktu	10	88	40	30	21	88	20	9	96	30	1	75	97	51	42
Task 4	Mesin	8	15	13	7	11	14	9	8	15	3	1	13	3	14	7
	Waktu	53	51	86	30	60	70	4	49	91	60	8	8	20	75	34
Task 5	Mesin	4	14	7	11	5	12	15	13	1	9	6	7	4	6	5
	Waktu	26	57	50	53	36	16	91	61	13	41	91	51	31	17	37
Task 6	Mesin	3	9	12	1	10	5	7	14	2	1	13	11	7	7	2
	Waktu	15	78	54	94	29	31	68	35	87	17	80	3	33	89	26
Task 7	Mesin	11	12	14	14	3	15	2	15	3	14	7	8	10	10	14
	Waktu	65	8	21	58	95	65	19	99	82	66	89	84	33	59	68
Task 8	Mesin	12	10	6	5	15	8	12	2	6	4	9	6	1	2	15
	Waktu	82	7	6	93	99	83	54	62	83	89	49	34	77	56	73
Task 9	Mesin	9	7	1	8	13	3	5	3	13	11	15	9	14	13	12
	Waktu	10	91	54	32	79	78	85	6	78	78	32	28	50	63	5
Task 10	Mesin	15	11	3	15	6	2	6	9	5	8	11	10	5	8	1
	Waktu	27	79	68	91	76	26	73	62	56	88	28	60	80	18	8
Task 11	Mesin	10	1	8	12	2	11	3	5	9	2	4	14	2	12	8
	Waktu	93	18	82	30	93	50	43	7	85	69	90	69	48	17	12
Task 12	Mesin	14	4	11	9	14	1	11	4	8	13	2	2	12	11	4
	Waktu	92	51	20	56	42	87	24	80	8	45	93	45	90	30	87
Task 13	Mesin	6	13	5	13	12	10	1	10	10	15	12	4	13	4	3
	Waktu	96	18	39	27	52	62	37	3	66	82	6	67	75	16	83
Task 14	Mesin	1	2	4	2	1	7	14	7	14	5	10	12	8	3	10
	Waktu	70	99	35	92	42	14	87	57	88	6	35	58	96	7	20
Task 15	Mesin	2	3	15	4	4	9	10	11	4	6	3	5	15	1	6
	Waktu	83	33	68	9	96	30	66	7	15	13	73	87	44	35	97

Gambar 3. Lama waktu penggunaan mesin setiap pekerja dalam menit

Setelah data di atas dimasukkan ke dalam program didapatkan keluaran berupa penjadwalan untuk menyelesaikan semua tugas dengan waktu tercepat. Keluaran tersebut dapat diilustrasikan sebagai berikut.



Gambar 4. *Job Shop Scheduling* dengan *backtracking*.

Apabila tidak menggunakan penjadwalan (brute force) waktu yang dibutuhkan adalah 11680 menit. Dengan menggunakan algoritma yang kami buat, seluruh pekerjaan dapat diselesaikan dengan waktu yang lebih baik yaitu selama 1662 menit. Sedangkan rekor penjadwalan terbaik dipegang oleh E. D. Taillard,

(1994) dengan algoritmanya *parallel taboo search techniques for the job shop scheduling problem* yang dibahas pada ORSA Journal on Computing 6, 108 117, dengan total waktu 1231 menit.

Dari hasil tersebut, dapat dilihat bahwa penjadwalan dengan *backtracking* dapat menghemat 85.77% waktu jika dibandingkan dengan algoritma *brute force*. Sedangkan penjadwalan dengan *parallel taboo search* dapat menghemat 89.46% waktu penyelesaian. Terlihat bahwa algoritma *backtracking* belum memberikan solusi yang optimal jika dibandingkan dengan *parallel taboo search*. Akan tetapi perbedaan persentase penghematan waktu tersebut tidaklah besar hanya berkisar 3.69%.

Dengan demikian, permasalahan *job shop scheduling* yang kami pilih dapat diselesaikan menggunakan *backtracking* dengan cukup baik jika dibandingkan tanpa memakai backtracking.

V. KESIMPULAN

Berdasarkan dari hasil penelitian dan diskusi yang kami lakukan, dalam permasalahan penjadwalan job shop, dapat disimpulkan bahwa penggunaan algoritma *backtracking* merupakan solusi yang tidak optimal namun mudah diimplementasikan dan hasil *output* yang didapat mendekati hasil *output* dari algoritma *parallel taboo search* yang merupakan algoritma yang paling optimal. Hal ini dapat dilihat dari selisih persentase penghematan waktu antara algoritma parallel *taboo search* dengan algoritma *backtracking* yaitu sebesar 3.69%. Sehingga dapat dikatakan penggunaan algoritma *backtracking* ini sangat memberikan hasil yang cukup optimal.

DAFTAR PUSTAKA

[1] Astuti M. 2013. Studi Penjadwalan Job Shop untuk meminimalkan Waktu Keseluruhan Menggunakan Pendekatan Algoritma Artificial Immune System. Sekolah Tinggi Teknologi Adisutjipto. Yogyakarta. Jurnal Angkasa. Vol 5 Hal 19.

[2] Firdaus. 2017. Implementasi penjadwalan Kuliah Job Shop Dengan Perancangan Jadwal Kuliah menggunakan Constraints Programming. STMIK Amik Riau. Pekan Baru. Jurnal & Penelitian Teknik Informatika. Vol 1 No 2 Hal 33.

[3] Johan, Adrianto, & Marsolim. 2006. Perancangan dan Implementasi Papan Jadwal Perkuliahan Berdasarkan Sistem Penjadwalan Otomatis. Jurnal Teknik Elektro (Tesla). Vol 8 Hal 75-95.

[4] Azanuddin, Purwadi, Zulkarnae I. 2017. Algoritma backtracking sebagai solusi game word search puzzle berbasis java mobile. *Jurnal SAINTIKOM*. 16(3): 297.

[5] Teneng, Purwadi, J., & Kurniawan, E. (2010). Penerapan algoritma backtracking pada permainan math maze. Jurnal Informatika. 6(1): 58.

[6] Apridiansyah Y & Rifqo MH. 2017. Implementasi algoritma backtracking dalam sistem informasi perpustakaan untuk pencarian judul buku (studi kasus unit pelayanan terpadu perpustakaan Universitas Muhammadiyah Bengkulu). *Jurnal Pseudocode*. IV(1): 93.

[7] Ploy, K & Mungwattana, A. 2010. Algorithm for Solving Job Shop Scheduling Problem Based on machine availability

[8] Chathurangi Shyalika. 2019. Job Shop Scheduling Problem (JSSP): An Overview. https://medium.com/datadriveninvestor/job-shop-scheduling-problem-jssp-an-overview-cd99970a02f8 (Diakses 12 Desember 2020)

[9] Gunawan, CA & Toba A. 2016. Pembangkitan Solusi Penjadwalan Berprioritas Melalui Penerapan Constraint Satisfaction Problem (Studi Kasus: Laboratorium Fakultas Teknologi Informasi Universitas XXX). *Jurnal Teknik Informasi dan Sistem Informasi*. Bandung[ID]: Universitas Kristen Maranatha. 2(1): 43.

[10] Hurink J, Jurisch B, Thole M. 1994. Tabu Search for the Job-Shop Scheduling Problem with Multi-purpose Machines. *OR Spektrum*. 15: 205-215

Lampiran

A. Source code

Source code dapat diakses melalui tautan berikut: ipb.link/aik2kel4