Algoritma Genetik Hibrida dalam Penyelesaian *Job-shop Scheduling Problem*

Samsu Sempena
Jurusan Teknik Informatika ITB, Bandung, email: if17088@students.if.itb.ac.id

Abstract *-- Makalah ini membahas mengenai algoritma genetik hibrida (dalam bahasa Inggris dikenal dengan Hybrid Genetic Algorithm) dalam menyelesaikan Job-shop Scheduling Problem (JSP) yaitu untuk menghasilkan jadwal yang optimal. JSP merupakan salah satu permasalahan optimasi kombinatorial tersulit dalam dunia informatika yang dikategorikan sebagai NP-Hard. Selama riset sekitar tiga puluh tahun terakhir belum ditemukan algoritma yang cukup efektif dalam menemukan solusi optimal untuk JSP karena algoritma yang tersedia masih memiliki kompleksitas eksponensial dan belum ada yang memiliki waktu asimptotik polinomial. Karena itu, untuk kasus penjadwalan dengan mesin dan job yang cukup banyak masih sangat sulit untuk diselesaikan. Namun, hingga saat ini algoritma-algoritma optimasi untuk menghasilkan solusi yang mendekati solusi optimal terus dikembangkan. Di antaranya adalah dengan teknik Priority Dispatch Rules, Bottleneck Based Heuristics, Local search methods and Meta-heuristics seperti Simulated-Annealing (SA) maupun Genetic Algorithm (GA). Sekalipun demikian teknik-teknik pendekatan yang ada masih belum cukup baik sehingga diajukanlah algoritma genetik hibrida yang menggabungkan Metode Heuristik dan Algoritma Genetik dengan harapan diperoleh algoritma yang lebih efektif.*

Kata Kunci*: Job Shop, Scheduling, NP-Hard, Genetic Algorithm, Hybrid.*

1. PENDAHULUAN

Job-shop Scheduling Problem merupakan suatu permasalahan untuk menentukan urutan operasi yang dilakukan dalam mesin yang ada dengan tujuan meminimumkan waktu proses total yang dibutuhkan. Diberikan sejumlah **m** *machine* (mesin) berbeda dan **n** *job* (pekerjaan) berbeda untuk dijadwalkan. Setiap pekerjaan terdiri dari sejumlah operasi yang harus dilakukan di suatu mesin tertentu selama durasi waktu tertentu. Ketika suatu operasi akan diproses di suatu mesin, ada kemungkinan operasi tersebut mengantri terlebih dahulu karena mesin tersebut sedang dipakai oleh operasi lain.

Ada beberapa ketentuan pada mesin dan pekerjaan, di antaranya :

- Suatu pekerjaan diproses hanya sekali dalam suatu mesin
- Suatu operasi tidak dapat diinterupsi
- Operasi dalam suatu pekerjaan harus diproses secara berurutan yaitu setelah operasi sebelumnya selesai dilakukan
- Setiap mesin hanya dapat memproses satu pekerjaan pada suatu waktu

Definisi formal :

- Himpunan pekerjaan $M = \{m_1, m_2, \ldots m_m\}$
- Himpunan mesin $J = \{ j_1, j_2, \ldots j_n\}$
- Himpunan operasi per job i $O_i = \{o_{i1}, o_{i2}, \ldots o_{im_i}\}$
- Tiap operasi memiliki waktu pemrosesan $\{\tau_{i1}, \tau_{i2}, \ldots \tau_{im_i}\}$
- Untuk setiap O terdapat himpuanan A , relasi biner urutan operasi. Jika $(v, w) \in A$, maka v harus dikerjakan sebelum w

Untuk setiap operasi v didefinisikan waktu mulai S(v)

$\forall v \in O: \quad S(v) \geq 0$

$\forall v, w \in O, (v, w) \in A: \quad S(v) + \tau(v) \leq S(w)$

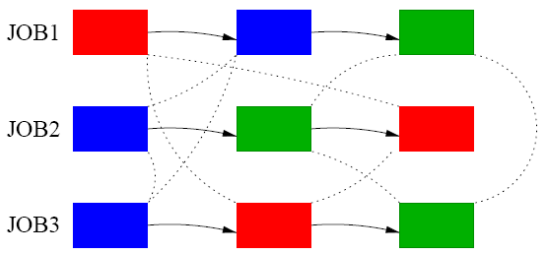
$\forall v, w \in O, v \neq w, M(v) = M(w): \; S(v) + \tau(v) \leq S(w) \; or \; S(w) + \tau(w) \leq S(v)$

Panjang suatu jadwal S adalah

$$len\,(S) = \max_{v \in O}(S(v) + \tau\,(v))$$

Contoh suatu persoalan JSP dalam dunia nyata adalah penjadwalan kegiatan di sekolah/universitas, penjadwalan perjalanan kereta api, penjadwalan penerbangan pesawat terbang,dsb.

Representasi JSP dalam graf terhubung untuk 3 mesin dan 3 job dengan masing-masing job memiliki 3 operasi adalah :



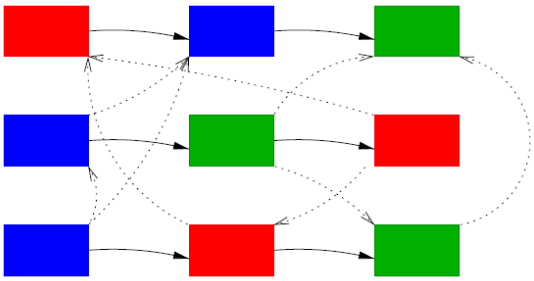
- Simpul (yaitu kotak berwarna biru,merah,dan hijau) menyatakan mesin tempat suatu operasi harus diproses.
- Sisi berarah menunjukkan urutan suatu operasi diproses dalam suatu job.
- Garis putus-putus (sisi tak berarah) menunjukkan bahwa kedua operasi berada dalam mesin yang sama dan salah satu dari kedua operasi itu harus mendahului yang lain.
- Bobot dari suatu sisi menyatakan durasi operasi sebelumnya diproses

JSP dengan **m** mesin dan **n** pekerjaan akan menghasilkan $(n!)^m$ jadwal, namun tidak semua jadwal tersebut valid. Suatu jadwal dikatakan valid apabila memnuhi kriteria berikut:

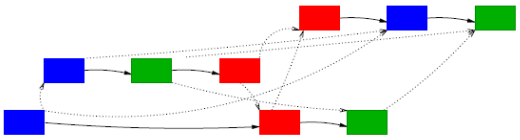
- Urutan pengerjaan operasi pada setiap job cocok dengan *routing* yang telah ditetapkan.
- Tidak ada *overlap* waktu pengerjaan pada operasi-operasi yang berpadanan dengan mesin yang sama.

Langkah-langkah untuk menentukan sebuah jadwal yaitu :

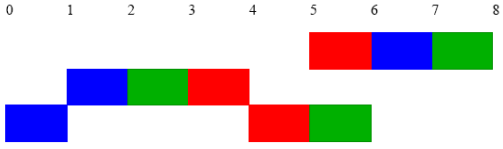
1. Menentukan arah untuk tiap sisi



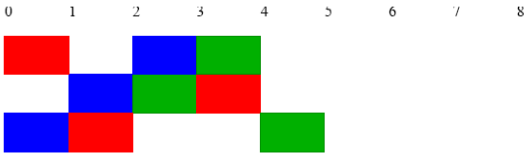
2. Mengurutkan graf tersebut sesuai urutan proses (dari kiri ke kanan)



3. Menuliskan titik awal dan akhir untuk tiap jadwal (misalkan durasi 1 satuan waktu untuk tiap operasi)



4. Namun, dapat diamati bahwa jadwal yang telah dihasilkan sebelumnya belum merupakan jadwal yang optimal dengan melihat jadwal lain yang dapat dibentuk dari kondisi seperti contoh di atas :



Menurut Lenstra et al (1977) untuk pasangan jumlah job dan mesin berukuran :

- 3*3
- N*2 dengan tidak lebih dari 3 operasi per job
- N*3 dengan tidak lebih dari 2 operasi per job
- N*3 dengan tiap operasi berdurasi 1 satuan unit waktu

Telah merupakan permasalahan NP Complete yaitu *non-deterministic polynomial time,* sebuah masalah yang tidak bisa diselesaikan dalam waktu yang polinomial (kompleksitasnya tidak polinomial)

Karena itu dibutuhkan suatu metode yang tepat untuk menyelesaikan JSP.

2. METODE PENYELESAIAN JSP

Dikarenakan topik utama dari makalah ini adalah untuk membahas penyelesaian JSP dengan algoritma genetik hibrida,maka cara-cara penyelesaian JSP yang telah ada hanya dibahas secara singkat. Sedangkan untuk algoritma genetik hibrida akan dibahas lebih lanjut di bagian selanjutnya.

2.1 Penyelesaian Optimum

2.1.1 Metode efisien

Menyelesaikan kasus penjadwalan dengan menghasilkan solusi optimal dengan algoritma yang efisien yaitu dalam kompleksitas polinomial untuk beberasa kasus JSP khusus.

Contohnya :

1. Johnson (1954) yang membuat algoritma yang efisien untuk menyelesaikan JSP untuk 2 mesin dan tiap job yang ada diproses di kedua mesin. Kasus JSP ini dinotasikan dengan $n/2/F/F_{max}$
2. Akers (1956) menyelesaikan kasus JSP berukuran 2xM dengan algoritma yang efisien
3. Jackson (1956) menyelesaikan JSP dengan ukuran Nx2 dengan tiap job memiliki paling banyak 2 operasi
4. Hefetz dan Adiri (1982) menyelesaikan JSP dengan ukuran Nx2 dengan tiap operasi memiliki durasi 1 unit satuan

waktu.

Namun, belum ada algoritma efisien untuk menyelesaikan JSP dengan N≥3 dan M≥3. Bahkan French (1982) memprediksikan bahwa tidak akan ada algoritma efisien untuk sebagian besar kasus JSP.

2.1.2 Formulasi Matematik

JSP ternyata dapat diselesaikan secara optimal dengan teknik pemrograman matematika dan formula matematika paling umum untuk JSP adalah *mixed linear programming* (MIP) oleh Manne (1960)

Minimise C_{max} *subject to :*

starting times $t_{ik} \geq 0$

precedence constraint $t_{ik} - t_{ih} \geq \tau_{ih}$

disjunctive constraint $t_{pk} - t_{ik} + \mathcal{K}(1-y_{ipk}) \geq \tau_{ik}$

$t_{ik} - t_{pk} + \mathcal{K}(y_{ipk}) \geq \tau_{pk}$

$\{i, p\} \in \mathcal{J} \qquad \{k, h\} \in \mathcal{M}$

if O_{ih} precedes O_{ik}

$y_{ipk} = 1$, if O_{ik} precedes O_{pk},

$y_{ipk} = 0$, otherwise

where $\mathcal{K} > \left(\sum_{i=1}^{n} \sum_{k=1}^{m} \tau_{ik} - min\left(\tau_{ik}\right) \right)$

Sekalipun elegan secara konseptual namun jumlah integer variable yang dibutuhkan meningkat secara eksponensial (Bowman 1959) dan formula yang lebih baik pun masih membutuhkan sejumlah bersar batasan. Sementara French (1982) mengatakan bahwa masalah penjadwalan dengan formulasi matematik tidak mungkin dikerjakan secara komputasi.

2.1.2 Teknik *Branch and Bound (BB)*

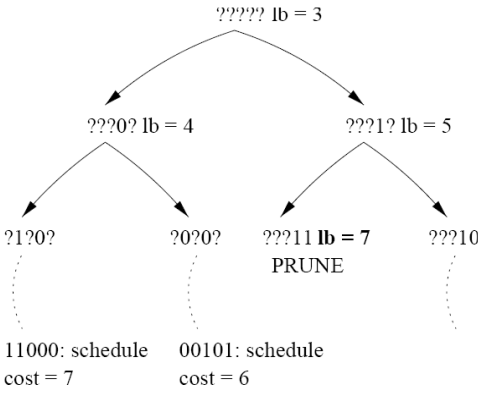
Teknik pencarian BB ini pertama dipelajari oleh Brooks dan White (1965), Ignall dan Schrage (1965), dan Lomnicki (1965).

McMahon dan Florian (1975) berhasil pertama kalinya mengaplikasikan BB pada satu mesin dekomposisi.

Langkah-langkahnya :

1. Pada pohon pencarian, berikan nama untuk setiap sisi tak berarah misalkan $x_1, x_2,..,x_n$ dengan $x_i \in \{0,1\}$
2. Setiap cabang diberikan nilai 0 dan cabang lain diberikan 1
3. Jika graf membentuk sirkuit atau estimasi batas bawah (*Lower Bound*) melebihi batas atas (*Upper Bound*) terbaik yang pernah didapat, maka subpohon dipangkas.
4. Setelah setiap sisi x_i memiliki nilai 0 atau 1 maka telah ditemukan suatu jadwal yang optimal.

Sekalipun penyelesaian optimal dengan BB cukup baik. Namun banyak peneliti yang memutuskan untuk beralih ke metode pendekatan (*approximation methods*) karena menganggap bahwa penyelesaian optimal tidaklah tepat untuk JSP.



Gambar pohon pembentukan jadwal dengan teknik *branch and bound*

2.2 Metode Pendekatan

Sekalipun metode pendekatan tidak menjamin solusi terbaik, namun mendekati solusi optimal, dalam waktu komputasi yang terjangkau dan karenanya lebih tepat untuk permasalahan dengan ukuran job dan mesin yang besar.

Beberapa metode pendekatan yang dikenal :

2.2.1 *Priority Dispatch Rules* (pdrs)

Prinsip pdrs adalah pada setiap langkah, setiap operasi yang dapat dijadwalkan diberikan suatu nilai prioritas dan operasi dengan prioritas tertinggi dipilih untuk diantrikan.

Peneliti mula-mula dari pdrs adalah Jackson (1955,1957), Smith (1956), Rowe dan Jackson (1956), Giffler dan Thompson (1960), dan Gere(1966).

2.2.2 *Bottleneck Based Heuristics*

Selama beberapa lama, metode approksimasi yang tersedia hanya pdrs namum Fisher dan Ronnooy Kan (1988) berhasil mengembangkan pendekatan yang lebih baik yang dikenal dengan Bottleneck Based Heuristics. Salah satu contohnya adalah *Shifting Bottleneck Procedure* (SBP)

oleh Adams et al(1988).

Langkah-langkah dalam SBP :
- Identifikasi submasalah
- Pemilihan *bottleneck*
- Menghasilkan solusi untuk submasalah
- Optimasi ulang jadwal

2.2.3 *Artificial Intelligence/AI* (Kecerdasan buatan) :
AI merupakan bagian dari sains computer yang mempelajari integrasi biologis dan kecerdasan komputer. Dua metodologi utama dalam AI yaitu :
- *Constraint Satisfaction* (CS)
- *Neural Network Methods*

2.2.4 *Local search methods and Meta-heuristics*
Dua buah jadwal dikatakan bertetangga jika jadwal yang lain dapat dibentuk dari suatu jadwal hanya dengan sedikit modifikasi pada grafnya.Mulai dari suatu jadwal secara acak kemudian mencari jadwal tetangganya yang lebih baik. Untuk metode ini akan dibahas lebih jelas di bagian selanjutnya.

Beberapa tipe *local search* :
- Problem space based methods
- Threshold Algorithms
- Genetic Algorithms (GAs)
- Tabu Search

3. ALGORITMA GENETIK HIBRIDA

Algoritma genetik hibrida merupakan gabungan dari algoritma genetik dengan pencarian lokal untuk menghasilkan solusi yang lebih optimal dengan waktu komputasi lebih singkat.

3.1 Algoritma Genetik

Algoritma genetik adalah metode adaptif yang dapat digunakan untuk menyelesaikan permasalahan pencarian dan optimasi. (Beasley et al (1993)). Algoritma ini didasarkan pada proses genetik pada organisme biologis. Setelah melewati banyak generasi, populasi berkembang sesuai dengan prinsip dari seleksi alam, yaitu *survival of the fittest.* Dengan meniru proses ini, algoritma genetik mampu mengembangkan solusi untuk permasalahan di dunia nyata jika algoritma ini disandikan secara tepat.

Sebelum algoritma genetik dapat dijalankan, sebuah jenis representasi yang tepat perlu ditentukan. Fungsi kecocokan yang memenuhi juga diperlukan, yang akan menilai solusi sementara yang dihasilkan. Selama algoritma dieksekusi, induk harus dipilih untuk reproduksi dan dikombinasikan untuk menghasilkan keturunan. Kromosom dari kedua orang tua dikombinasikan, umumnya menggunakan mekanisme *crossover* dan *mutation (*mutasi). Mutasi umumnya diaplikasikan pada individu untuk memastikan keberagaman populasi.

Di bawah ini adalah pseudocode algoritma genetik yang sering dipakai.

Genetic Algorithm
{
  Generate initial population ***P***t
  Evaluate population ***P***t
  While stopping criteria not satisfied **Repeat**
  {
    Select elements from ***P***t to copy into ***P***t+l
    Crossover elements of ***P***t and put into ***P***t+l
    Mutation elements of ***P***t and put into ***P***t+l
    Evaluate new population ***P***t+l
    Pt = ***P***t+l
  }
}

3.1.1 Representasi Kromosom dan Penyandiannya

Algoritma genetik yang dideskripsikan dalam makalah ini menggunakan kunci acak alphabet **U**(0,1) dan strategi evolusi yang sama dengan diajukan oleh Bean (1994). Kegunaan penting dari kunci acak yaitu setiap generasi yang dihasilkan oleh crossover adalah solusi yang memenuhi kriteria jadwal yang valid.

Sebuah kromosom merepresentasikan solusi dari JSP dan disandikan sebagai vector dari kunci acak (*random keys*). Setiap kromosom solusi dibuat dari 2n gen dengan n menyatakan jumlah operasi.

$$Chromosome = (gene_1, gene_2, ..., gene_n, gene_{n+1}, ..., gene_{2n}).$$
N gen pertama digunakan sebagai prioritas operasi.

$$Priority_j = Gene_j$$
Gen di antara n+1 dan 2n digunakan untuk menentukan waktu tunda yang digunakan saat menjadwalkan operasi.

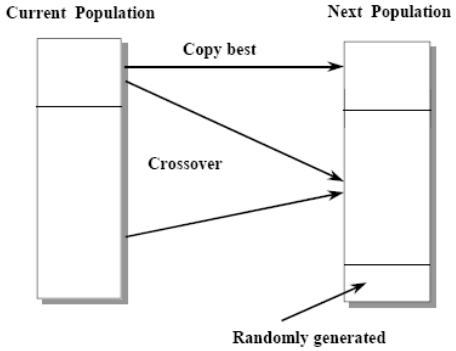
$$Delay_g = gene_g \cdot 1.5 \cdot MaxDur$$
MaxDur adalah maksimum durasi dari semua operasi sedangkan faktor pengali 1,5 didapat dari eksperimen.

3.1.2 Strategi Evolusi

Untuk menghasilkan solusi yang baik, kunci acak dari vektor populasi dioperasikan oleh algoritma genetik. Reproduksi dan *cross over* menentukan induk mana yang akan memiliki keturunan, meningkatkan kualitas populasi, dan memusatkan perkembangannya. Mutasi memungkinkan variasi dan menggantikan materi genetik yang hilang dalam reproduksi dan *crossover.*

Reproduksi dilakukan dengan menyalin sebagian dari individual terbaik ke generasi selanjutnya sehingga

solusi semakin membaik dari satu generasi ke selanjutnya. Masalahnya adalah jika populasi mengerucut(memusat) ke minimum lokal, namun ini dapat diatasi dengan laju mutasi yang tinggi.



Proses transisi antara generasi

3.2 Prosedur Menghasilkan Jadwal

Untuk setiap perulangan **g**,terdapat waktu penjadwalkan $\mathbf{t_g}$. Set yang aktif A_g terdiri dari

$A_g(t_g) = \{j \in J \mid F_j - d_j \leq t_g < F_j\}$
Kapasitas mesin yang tersisa pada t_g adalah

$$RMC_m(t_g) = 1 - \sum_{j \in A_g} r_{j,m}$$

S_g terdiri dari semua operasi yang telah dijadwalkan sampai g. F_g terdiri dari waktu selesai dari operasi dalam S_g. $Delay_g$ adalah waktu tunda pada g. Eg terdiri dari semua operasi yang sebelumnya dapat dilakukan dalam interval $[t_g, t_g+Delay_g]$.

$E_g(t_g, Delay_g) = \{ j \in J \backslash S_g \mid F_i \leq t_g + Delay_g \ (I \in P_j)\}$

Pseudocode untuk prosedur menghasilkan jadwal adalah :

Inisialisasi : g = 0; t_g = 0; $A_0 = \{0\}$; RMCm (0) = 1; $F_g(0) = \{0\}$; $S_g(0) = \{0\}$

While $|S_g| \leq n+1$ repeat
{
penambahan pengulangan
g = g+1

menentukan waktu yang berasosiasi dengan g
$t_g = Min_{j \in A_g} F_j$

Menghitung $A_g(t_g)$, $RMC_m(t_g)$,E_g=Eg(t_g, $Delay_g$)

While $E_g \neq \{\}$ **repeat**
{
Pilih operasi dengan prioritas tertinggi
j* = argmax{$PRIORITY_j$} $(j \in E_g)$

hitung waktu selesai tersingkat (berdasarkan urutan saja)
$EF_{j*} = max_{i \in P_j}\{F_i\} + d_j$

Hitung waktu selesai tersingkat (berdasarkan urutan dan kapasitas)

$$F_{j^*} = \min\left\{ t \in \left[EF_{j^*} - d_{j^*}, \infty \right] \cap F_g \mid r_{j^*,m} \leq RMC_m(\tau), \ m \mid r_{j^*,m} > 0, \ \tau \in \left[t, t + d_{j^*} \right] \right\} + d_{j^*}$$

Penambahan pengulangan
g = g + 1

hitung $A_g(t_g)$, $RMC_m(t_g)$,E_g=Eg(t_g, $Delay_g$)

perbaharui $A_g(t_g)$,$RMC_m(t_g)$,E_g=Eg(t_g, $Delay_g$)

perbaharui S_g dan F_g
$S_g = S_{g-1} \cup \{j^*\}$
$F_g = F_{g-1} \cup \{F_j^*\}$
}
}
Menghitung waktu rentang proses
$F_{n+1} = Max_{i \in P_{n+1}}\{F_i\}$

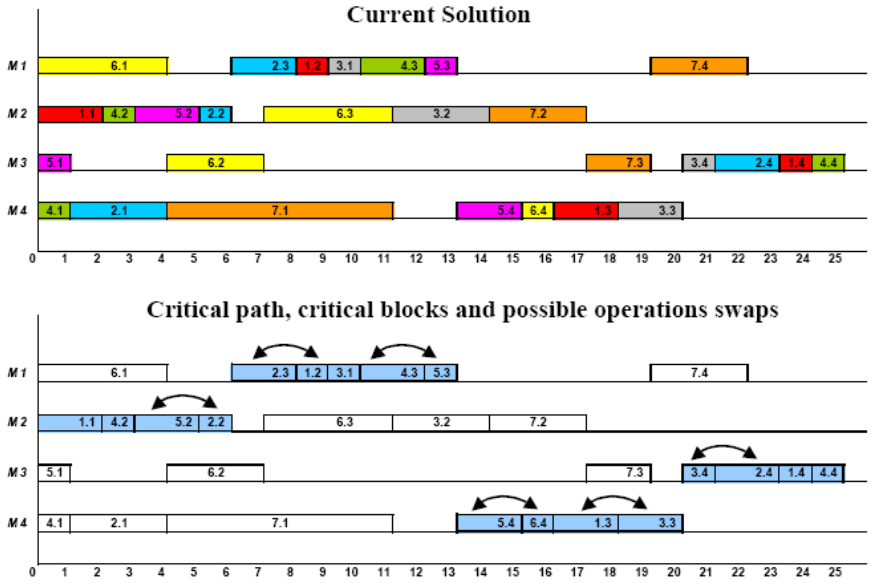
3.3 Prosedur Pencarian Lokal (*Local Search*)

Karena tidak ada kepastian bahwa jadwal yang dihasilkan dalam fase pembentukan merupakan optimal lokal dibandingkan dengan tetangga-tetangganya, pencarian lokal dapat diaplikasikan untuk mencoba meminimalkan waktu rentang proses.

Prosedur pencarian lokal dimulai dengan mengidentifikasi daerah kritis dari solusi yang didapat dari prosedur menghasilkan jadwal. Setiap operasi dalam daerah kritis disebut operasi kritis. Dimungkinkan untuk mendekomposisikan daerah kritis menjadi sejumlah blok , di mana sebuah blok adalah maksimal urutan dari operasi kritis yang berdekatan yang membutuhkan mesin yang sama untuk diproses.

Dalam makalah ini, pendekatan yang digunakan adalah Nowicki dan Smutnicki (1996). Dalam pendekatan ini jika sebuah pekerjaan dan mesin dari operasi sebelumnya dari operasi kritis juga termasuk operasi kritis maka dipilih operasi yang lebih dulu muncul dalam urutan operasi.

Jika penukaran yang dilakukan memperkecil waktu rentang proses, maka disetujui. Jika tidak maka penukaran dibatalkan. Pada saat terjadi penukaran, daerah kritis mungkin berubah dan daerah kritis yang baru perlu diproses kembali. Jika tidak ada penukaran yang memperkecil waktu rentang proses, maka pencarian lokal selesai.



Ketetanggaan Nowicki dan Smutnicki (1996)

Pseudocode untuk pencarian lokal :

```
Local_Search (CurrentSolution)
do
{
  CurrentSolutionUpdated = False
  Determine the critical path and all critical blocks of  CurrentSolution
  while Unprocessed blocks and not CurrentSolutionUpdated do
  {
     if not First Critical Block then
       NewSolution := Swap first two operations of block in CurrentSolution
       if Makespan ( NewSolution ) < Makespan ( CurrentSolution ) then
            CurrentSolution = NewSolution
          CurrentSolutionUpdated = true
       endif
     endif
     if not Last Critical Block and not CurrentSolutionUpdated then
          NewSolution = Swap last two operations of block in CurrentSolution
          if Makespan ( NewSolution ) < Makespan ( CurrentSolution ) then
               CurrentSolution = NewSolution
               CurrentSolutionUpdated = true
        endif
     endif
  }
}
until CurrentSolutionUpdated = false
return CurrentSolution
```

4. HASIL DAN PEMBAHASAN

Teknik pendekatan algoritma genetik hibrida yang dibahas dalam makalah ini telah dibandingkan dengan teknik pendekatan lainnya untuk mengetahui keefektiannya. Adapun 43 contoh masalah yang diujikan diambil dari 2 tes standar untuk JSP yaitu Fischer dan Thompson (1963) soal nomor FT06, FT10, dan FT20, juga dari Lawrence (1984) soal nomor LA01 sampai LA40.

Hasil perbandingannya dapat dilihat dari tebel berikut:

Instance	Size	BKS	HGA	Time (sec.)	Wang and Zheng (2001)	Aiex et al. (2001)	Binato et al. (2002)	Nowicki and Smutnicki (1996)	Gonçalves and Beirão (1999)	Croce et al.. (1995)	Dorndorf & Pesch			Aarts et al.		Storer et al. (1992)
											P-GA (1995)	SBGA (40) (1995)	SBGA (60) (1995)	GLS1 (1994)	GLS2 (1994)	
FT06	6x6	55	55	13	55	55	55	55	55		-					
FT10	10x10	930	930	292	930	930	938	930	936	946	960			935	945	952
FT20	20x5	1165	1165	204	1165	1165	1169	1165	1177	1178	1249			1165	1167	
LA01	10x5	666	666	37	666	666	666	666	666	666	666	666		666	666	666
LA02	10x5	655	655	51		655	655	655	666	666	681	666		668	659	
LA03	10x5	597	597	39		597	604	597	597	666	620	604		613	609	
LA04	10x5	590	590	42		590	590	590	590	-	620	590		599	594	
LA05	10x5	593	593	32		593	593	593	593	-	593	593		593	593	
LA06	15x5	926	926	99	926	926	926	926	926	926	926	926		926	926	
LA07	15x5	890	890	86		890	890	890	890	-	890	890		890	890	
LA08	15x5	863	863	99		863	863	863	863	-	863	863		863	863	
LA09	15x5	951	951	94		951	951	951	951	-	951	951		951	951	
LA10	15x5	958	958	91		958	958	958	958	-	958	958		958	958	
LA11	20x5	1222	1222	197	1222	1222	1222	1222	1222	1222	1222	1222		1222	1222	
LA12	20x5	1039	1039	201		1039	1039	1039	1039	-	1039	1039		1039	1039	
LA13	20x5	1150	1150	189		1150	1150	1150	1150	-	1150	1150		1150	1150	
LA14	20x5	1292	1292	187		1292	1292	1292	1292	-	1292	1292		1292	1292	
LA15	20x5	1207	1207	187		1207	1207	1207	1207	-	1237	1207		1207	1207	
LA16	10x10	945	945	232	945	945	946	945	977	979	1008	961	961	977	977	981
LA17	10x10	784	784	216		784	784	784	787	-	809	787	784	791	791	794
LA18	10x10	848	848	219		848	848	848	848	-	916	848	848	856	858	860
LA19	10x10	842	842	235		842	842	842	857	-	880	863	848	863	859	860
LA20	10x10	902	907	235		902	907	902	910	-	928	911	910	913	916	
LA21	15x10	1046	1046	602	1058	1057	1091	1047	1047	1097	1139	1074	1074	1084	1085	
LA22	15x10	927	935	629		927	960	927	936	-	998	935	936	954	944	
LA23	15x10	1032	1032	594		1032	1032	1032	1032	-	1072	1032	1032	1032	1032	
LA25	15x10	977	986	609		984	1028	977	1004	-	1014	1008	1007	1016	1010	
LA26	20x10	1218	1218	1 388	1218	1218	1271	1218	1218	1231	1278	1219	1218	1240	1236	
LA27	20x10	1235	1256	1 251		1269	1320	1236	1260	-	1378	1272	1269	1308	1300	
LA28	20x10	1216	1232	1 267		1225	1293	1216	1241	-	1327	1240	1241	1281	1265	
LA29	20x10	1157	1196	1 350		1203	1293	1160	1190	-	1336	1204	1210	1290	1260	
LA30	20x10	1355	1355	1 260		1355	1368	1355	1356	-	1411	1355	1355	1402	1386	
LA31	30x10	1784	1784	3 745	1784	1784	1784	1784	1784	1784	-			1784	1784	
LA32	30x10	1850	1850	3 741		1850	1850	1850	1850	-	-			1850	1850	
LA33	30x10	1719	1719	3 637		1719	1719	1719	1719	-	-			1719	1719	
LA34	30x10	1721	1721	3 615		1721	1753	1721	1730	-	-			1737	1730	
LA35	30x10	1888	1888	3 716		1888	1888	1888	1888	-	-			1894	1890	
LA36	15x15	1268	1279	1 826	1292	1287	1334	1268	1305	1305	1373	1317	1317	1324	1311	1305
LA37	15x15	1397	1408	1 860		1410	1457	1407	1441	-	1498	1484	1446	1449	1450	1458
LA38	15x15	1196	1219	1 859		1218	1267	1196	1248	-	1296	1251	1241	1285	1283	1239
LA39	15x15	1233	1246	1 869		1248	1290	1233	1264	-	1351	1282	1277	1279	1279	1258
LA40	15x15	1222	1241	2 185		1244	1259	1229	1252	-	1321	1274	1252	1273	1260	1258

Tabel 1. Hasil Eksperimen Penyelesaian JSP dengan Berbagai Algoritma

Ket : BKS = Best Known Solution (solusi terbaik yang telah ditemukan)
HGA = Hybrid Genetic Algorithm (algoritma genetik hibrida)

Algoritma	Jumlah kasus Terselesaikan	Deviasi dari solusi terbaik		Peningkatan AGH
		AL	AGH	
Problem and Heuristic Space				
Storer et al. (1992)	11	2,44%	0,56%	1,88%
Genetic Algorithms				
Aarts et al. (1994) - GLS1	42	1,97%	0,40%	1,57%
Aarts et al. (1994) - GLS2	42	1,71%	0,40%	1,31%
Croce et al (1995)	12	2,37%	0,07%	2,30%
Dorndorf et al. (1995)- PGA	37	4,61%	0,46%	4,15%
Dorndorf et al. (1995)- SBGA(40)	35	1,42%	0,48%	0,94%
Dorndorf et al. (1995)- SBGA(60)	20	1,94%	0,84%	1,10%
Goncalves and Beirao (1999)	43	0,90%	0,39%	0,51%
GRASP				
Binato et al. (2002)	43	1,77%	0,39%	1,38%
Aiex et al. (2001)	43	0,43%	0,39%	0,04%
Hybrid Genetic and Simulated Annealing				
Wang and Zheng (2001)	11	0,28%	0,08%	0,20%
Tabu Search				
Nowicki dan Smutnicki (1996)	43	0,05%	0,39%	-0,34%

Tabel 2 Deviasi Solusi yang Dihasilkan Algoritma JSP

Ket : AL = Algoritma Lain
AGH = Algoritma Genetik Hibrida

Secara keseluruhan Algoritma Genetik Hibrida berhasil menyelesaikan ke-43 tes yang tersedia dan mendapatkan nilai deviasi rata-rata 0.39%. Algoritma Genetik Hibrida menggungguli hampir semua algoritma yang lain, dengan perkecualian Algoritma *Tabu Search* Nowicki dan Smutnicki yang memiliki performa lebih baik terutama dalam kasus 15x15.

5. KESIMPULAN

Jobshop Scheduling Problem merupakan masalah yang masuk kelompok NP-Hard oleh karena belum adanya algoritma eksak yang mampu menyelesaikan kasus JSP dalam kompleksistas polinomial. Dua metode utama dalam menyelesaikan JSP yaitu penyelesaian optimum dan metode pendekatan. Metode pendekatan dianggap lebih tepat untuk menyelesaikan JSP selama belum ditemukannya algoritma dalam waktu asimptotik polinomial untuk JSP mengingat deviasi dari solusi optimum yang kecil dan waktu komputasi yang lumayan cepat.

Algoritma Genetik Hibrida merupakan metode pendekatan yang diajukan dengan harapan diperolehnya suatu cara penyelesaian JSP yang lebih optimal daripada kedua algoritma (heuristik dan genetik) digunakan secara terpisah. Terbukti bahwa algoritma ini lebih baik dari banyak algoritma pendekatan lainnya.

DAFTAR REFERENSI

[1] Bean, J.C., (1994). Genetics and Random Keys for Sequencing and Optimization, ORSA Journal on Computing, Vol. 6, pp. 154-160.

[2] Fisher, H. and Thompson, G.L., (1963). Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, in: Industrial scheduling, J.F. Muth and G.L. Thompson (eds.), Prentice-Hall, Englewood Cliffs, NJ, pp. 225-251.

[3] Gonçalves, J.F. and Mendes, (2002). A Hybrid Genetic Algorithm for the Jobshop Scheduling Problem.

[4] Jain, A.S. and Meeran, S. (1999). A State-of-the-Art Review of Job-Shop Scheduling Techniques. European Journal of Operations Research, Vol. 113, pp. 390-434.

[5] Lenstra, J. K. and Rinnooy Kan, A. H. G. (1979) Computational Complexity of Discrete Optimization Problems, *Annals of Discrete Mathematics*, vol 4, 121-140..

[6] Nowicki, E. and Smutnicki, C. (1996). A Fast Taboo Search Algorithm for the Job-Shop Problem, Management Science, Vol. 42, No. 6, pp. 797-813.