# PENYELESAIAN *PERMUTATION FLOW SHOP SCHEDULING PROBLEM* DENGAN MENGGUNAKAN ALGORITMA MEMETIKA DAN GRASP


## ABSTRAK

*Permutation Flowshop Scheduling Problem (PFSP) adalah masalah penjadualan yang berkaitan dengan pengurutan pemrosesan pekerjaan pada mesin. Setiap pekerjaan harus diproses tepat satu kali pada setiap mesin dalam urutan yang sama dengan waktu proses tertentu dan pekerjaan diproses dalam urutan yang sama pada setiap mesin. Tujuan penulisan ini adalah melihat kinerja kombinasi Algoritma Memetika (AM) dan metode Greedy Randomized Adaptive Search Procedure (GRASP) dalam menyelesaikan PFSP dengan tujuan meminimumkan makespan. Kinerja metode AM dan GRASP dilihat dari kedekatan solusi yang dihasilkan dengan Best Known Solution (BKS) pada Taillard's Benchmark dan dari waktu komputasinya.*



***Kata Kunci**: Algoritma Memetika, GRASP, Iterated Local Search, Path Relinking, Permutation Flowshop Scheduling Problem.*



**Nola Marina**

Universitas Gunadarma

nola.marina@staff.gunadarma.ac.id


## PENDAHULUAN

Di bidang industri, keuangan, perdagangan dan berbagai aktivitas lain dibutuhkan perencanaan yang baik paling tidak untuk memperoleh keuntungan yang tinggi, kualitas yang baik, biaya rendah, atau efisiensi sumber daya. Masalah penjadualan menjadi aspek yang sangat penting, karena penjadualan merupakan salah satu elemen perencanaan.

*Shop Scheduling Problem* (SP) adalah masalah penjadualan yang berkaitan dengan pengurutan pemrosesan sejumlah pekerjaan pada sejumlah mesin. SP merupakan masalah optimisasi yang memiliki karakteristik antara lain terdiri dari $m$ mesin dan $n$ pekerjaan. Setiap pekerjaan harus diproses pada setiap mesin. Masing-masing mesin dapat memroses paling banyak satu pekerjaan pada suatu waktu. Setiap pekerjaan harus diproses pada suatu mesin selama suatu periode waktu tertentu tanpa interupsi. Setiap pekerjaan hanya dapat diproses oleh satu mesin dalam satu waktu.

*Flow Shop scheduling Problem* (FSP) adalah SP dengan syarat tambahan setiap pekerjaan harus diproses tepat satu kali pada setiap mesin, di mana urutan mesin yang dilalui setiap pekerjaan harus sama. Sedangkan *Permutation Flowshop Scheduling Problem* (PFSP) adalah kasus khusus dalam FSP dengan $n$ pekerjaan diproses dalam urutan yang sama pada setiap mesin. Maka kandidat solusi dari PFSP adalah jadual yang direpresentasikan oleh urutan/barisan $n$ pekerjaan. Ruang solusi untuk PFSP adalah sebesar $n!$.

Beberapa asumsi yang dipakai dalam PFSP adalah jumlah pekerjaan $n$ dan jumlah mesin $m$ berhingga, setiap pekerjaan *independent* satu sama lain, waktu proses pekerjaan $i$ pada mesin $j$ diberikan, setiap mesin dalam keadaan siap dipakai dan selalu ada selama periode penjadualan, waktu *set up* termasuk dalam waktu proses, dengan kata lain diabaikan, sumber daya selalu tersedia selama periode penjadwalan

Tujuan dari PFSP adalah menemukan urutan $n$ pekerjaan yang meminimumkan total waktu penyelesaian semua pekerjaan atau *makespan*. PFSP termasuk kelas *NP-Hard* jika mesin yang digunakan lebih dari 3 [RAV06]. Metode eksak tidak mampu menyelesaikan masalah ini, kecuali masalah ukuran kecil. Masalah PFSP lebih efektif diselesaikan dengan menggunakan metode heuristik.

Metode heuristik yang digunakan dalam penelitian-penelitian terkait PFSP antara lain Metode NEH oleh Nawaz, Enscore, and Ham 1983, *Simulated Annealing* oleh Ogbu dan Smith 1990, *Tabu search* oleh Taillard 1990, Algoritma Genetika oleh Murata et al 1996, *Hybrida Genetic Algorithm* (HGA) oleh El Bouri, Algoritma *Genetic Local Search* (GLS) oleh Yamada et al, 1997, Algoritma Robust Genetik (RGA) oleh Ruiz et al, *Iterated Local Search* (ILS)oleh Thomas Stutzle 1998, *Greedy Randomized Adaptive Search Procedure* (GRASP), dan Algoritma Memetika (AM) oleh Pablo Moscato.

## METODE PENELITIAN

Di sini PFSP akan diselesaikan dengan menggunakan kombinasi AM dengan metode GRASP. Biasanya, populasi awal pada AM dibangun secara acak sehingga AM memerlukan iterasi yang cukup banyak untuk konvergen ke solusi optimal. Pada metode kombinasi ini diusulkan untuk membangun populasi awal yang cukup baik, sehingga AM hanya memerlukan operator yang sederhana dan iterasi yang sedikit untuk konvergen lebih cepat ke solusi optimal. Ide utamanya adalah metode GRASP membentuk kumpulan solusi yang cukup baik terlebih dahulu, untuk kemudian digunakan oleh AM sebagai populasi awal.

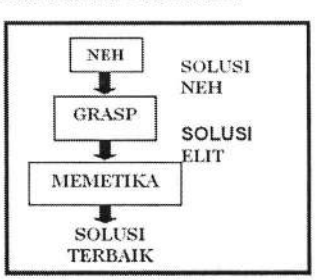
AM merupakan penggabungan dari Algoritma Genetika dengan *Local Search*. Algoritma Genetika (AG) adalah metode heuristik yang banyak digunakan untuk menyelesaikan masalah optimisasi yang berdasarkan proses evolusi biologi. AG diajukan oleh John Holland pada tahun 1975. Kandidat solusi dari suatu masalah direpresentasikan sebagai sekumpulan gen yang disebut kromosom.

AG diawali dengan pembentukan kumpulan koromosom yang disebut populasi. Masing-masing kromosom pada populasi akan dievaluasi menggunakan fungsi *fitness*. Kemudian secara iteratif akan dibentuk populasi baru dengan menerapkan operator-operator genetika hingga mencapai kriteria berhenti. Operator dasar pada AG adalah seleksi, *crossover*, dan mutasi. *Local search* pada AM bertujuan untuk melakukan perbaikan lokal yang dapat diterapkan sebelum dan atau sesudah proses seleksi, *cross over* dan mutasi.

GRASP adalah metode heuristik yang menggunakan proses iteratif dalam pencarian solusi. Ide dari GRASP adalah setiap iterasinya terdiri dari 2 tahap, yaitu pembentukan solusi awal dan peningkatan kualitas solusi dengan menerapkan metode ILS dan PR. Solusi yang memenuhi kriteria pada setiap iterasi, dimasukkan ke dalam kumpulan solusi yang disebut 'solusi elit' atau *Pool*.

*Iterated Local Search* (ILS) adalah metode heuristik yang secara iteratif melakukan perbaikan terhadap solusi sekarang dengan cara menerapkan *Local Search* terhadap modifikasi dari solusi sekarang. Sedangkan *Path Relinking* (PR) adalah metode heuristik untuk meningkatkan kualitas solusi dengan cara mengeksplorasi subruang solusi atau *path* di antara dua solusi yang baik yang sudah ada.

Prosedur AM dan GRASP dalam menyelesaikan PFSP dapat dijelaskan melalui Gambar 1 berikut ini.




**Gambar 1. *Flowchart* Metode AM dan GRASP**

**GRASP dalam Menyelesaikan PFSP**

Pada metode GRASP digunakan metode NEH untuk membentuk solusi awal sebelum masuk ke proses iterasi. Solusi dari metode NEH dimasukkan ke dalam kumpulan solusi atau *Pool*. Kemudian dilakukan LS terhadap solusi NEH, dan solusi LS yang dihasilkan dimasukkan ke dalam *Pool* dengan syarat solusi tersebut tidak terlalu mirip dengan solusi yang sudah ada dalam *Pool*.

Pada Algoritma GRASP standar, pada setiap iterasi dilakukan pembentukan solusi baru. Sedangkan pada metode GRASP kita menjaga suatu kumpulan solusi dengan kualitas yang baik, untuk kemudian diterapkan prosedur *Path Relinking* (PR) pada kumpulan solusi tersebut.

Proses iterasi pada metode ini menggunakan metode ILS dan PR untuk perbaikan solusi, dengan prosedur sebagai berikut:

1. Solusi awal: solusi awal yang digunakan adalah *Current Solution*. Untuk iterasi pertama, *Current Solution* adalah solusi terbaik dari *Pool*. Untuk iterasi selanjutnya, *Current Solution* selalu di-*update* dengan solusi hasil ILS dan PR.
2. *Perturbation:* dilakukan terhadap solusi awal dengan melakukan *swap* terhadap 2 posisi yang dipilih secara acak.
3. *Local Search:* diterapkan pada solusi hasil dari *perturbation*, dengan prosedur *insertion neighborhood*. Prosedur *insertion* pada *insertion neighborhood* di sini adalah *shift insertion*. Solusi *Local Search* kemudian dimasukkan ke *Pool* dengan syarat tidak terlalu mirip dengan solusi yang sudah ada dalam *Pool* dan *makespan* nya lebih kecil daripada solusi terbaik.
4. PR: diterapkan pada 2 solusi dalam *Pool*, yaitu solusi terbaik dan solusi yang dipilih acak dari *Pool*. PR tidak dilakukan pada setiap iterasi, tetapi dengan frekuensi tertentu yang disebut *Path Relinking Frequency* (PRF). Solusi PR dimasukkan ke dalam *Pool* dengan syarat tidak terlalu mirip dengan solusi yang sudah ada dalam *Pool* dan *makespan*-nya lebih kecil daripada solusi terbaik
5. Kriteria penerimaan solusi PR: jika ditemukan peningkatan kualitas solusi, *Current Solution* diganti dengan solusi PR. Jika tidak ditemukan peningkatan, *Current Solution* diganti dengan solusi PR dengan probabilitas $p_T > 0$. Jika $n$ adalah jumlah pekerjaan, $m$ adalah jumlah mesin, dan $p_{ij}$ adalah waktu proses pekerjaan $i$ pada mesin $j$, maka *temperature* T adalah

$$T = 0.5 \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}}{n.m.10}$$

Probabilitas penerimaan solusi adalah:

$p_T$ = *exp* (- *makespan* solusi PR *makespan Current solution*) / T

6. Kriteria penerimaan solusi *Local Search:* jika ditemukan peningkatan kualitas solusi, *Current Solution* diganti dengan solusi *Local Search*. Jika tidak ditemukan peningkatan, *Current Solution* diganti dengan solusi *Local Search* dengan probabilitas $p_T > 0$.

$p_T$ = *exp*(- *makespan* solusi LS - *makespan Current solution*) / T

Hasil dari metode GRASP adalah kumpulan solusi elit, yang selanjutnya akan digunakan oleh AM sebagai populasi awal.

**AM dalam Menyelesaikan PFSP**

Setelah pembentukan populasi awal, semua solusi pada populasi diperbaiki dengan menggunakan *Local Search* terlebih dahulu sebelum masuk ke proses evolusi. Pada proses evolusi, kromosom orangtua diseleksi dengan menggunakan *Roulette wheel selection*, untuk dikawin-silangkan menggunakan operator *Path Crossover*. Setelah itu dilakukan mutasi terhadap solusi terbaik dalam populasi dan *Local Search* diterapkan lagi pada populasi yang telah dihasilkan. Kemudian populasi sekarang digantikan oleh populasi baru hasil evolusi.

Prosedur ini dilakukan terus-menerus pada setiap generasi. Jika sampai sejumlah generasi tertentu tidak terjadi peningkatan kualitas populasi secara berturut-turut, maka semua solusi pada populasi diganti dengan menggunakan *Cold Restart*. Setelah penggantian seluruh solusi dalam populasi, prosedur AM dilanjutkan hingga mencapai kriteria berhenti.

Kriteria berhenti yang digunakan adalah maksimum generasi. Komponen utama AM yang digunakan adalah pembentukan populasi awal, *long search*, evaluasi nilai *fitness*, *Roulette Wheel selection*, *Path Crossover* (PX), mutasi, dan *cold restart*.

Populasi awal yang digunakan adalah kumpulan solusi atau *Pool* yang dihasilkan pada metode GRASP sebelumnya, yaitu sebanyak 15 solusi terbaik, ditambah sejumlah solusi yang dibangun secara random hingga jumlahnya sesuai *popsize* yang diinginkan.

Pada AM untuk menyelesaikan PFSP dalam tulisan ini, *Local Search* dilakukan terhadap semua solusi pada populasi awal dan setelah proses mutasi pada setiap iterasi dengan probabilitas yang relatif kecil.

Nilai *fitness* yang digunakan pada AM untuk menyelesaikan PFSP adalah

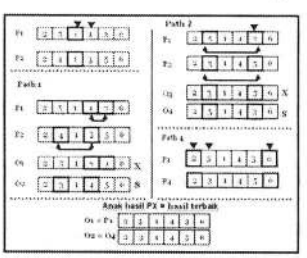
$$fitness_i = (f_w - f_i) + c$$

Di mana $f_w$ adalah nilai *makespan* terbesar dalam populasi, $f_i$ adalah nilai *makespan* kromosom $I$, dan $c$ adalah bilangan kecil sebagai nilai *fitness* terkecil yang diinginkan.

Pemilihan orangtua pada AM dilakukan dengan menggunakan *Roulette wheel selection*. Metode ini menggunakan prinsip pada permainan *roulette wheel*. Masing-masing kromosom menempati potongan lingkaran pada *roulette wheel* dengan ukuran yang proporsional, sebanding dengan nilai *fitness*nya. Pemilihan kromosom dilakukan dengan cara memutar *roulette wheel*. Semakin besar nilai *fitness* suatu kromosom, maka semakin besar kemungkinan kromosom itu untuk terpilih. Namun tidak tertutup kemungkinan komosom dengan nilai *fitness* yang kecil bisa terpilih.

Prosedur PX mirip dengan *path relinking*. Kelebihan PX dibandingkan dengan operator *crossover* biasa adalah PX menjamin kromosom anak tidak akan lebih buruk daripada kromosom orangtua. Prosedur PX adalah sebagai berikut:

- Pilih 2 kromosom orangtua, kemudian pencarian dilakukan dengan cara membentuk path antara kedua orangtua.
- Dimulai dari posisi random sampai mencapai posisi ini kembali.
  *Allele* dari kedua orangtua pada posisi random dibandingkan. Jika berbeda, lakukan *swap* antara *allele* yang berbeda tersebut pada masing-masing kromosom orangtua, sehingga dihasilkan 2 kromosom anak. Hal ini dilakukan untuk posisi selanjutnya dengan prosedur yang sama, hingga sampai ke posisi random yang dipilih pertama kali.
- Pada setiap *path*, algoritma ini memilih kromosom hasil *swap* dengan *makespan* yang lebih kecil daripada kromosom orangtua,. Kromosom anak hasil dari *crossover* ini adalah solusi terbaik yang ditemukan selama membentuk *path*.



**Gambar 2 Contoh prosedur *Path Crossover***

Mutasi disini bekerja seperti *perturbation* pada GRASP. Jenis operator mutasi yang digunakan adalah *Exchange mutation*, yaitu *swap* 2 *allel* yang dipilih secara acak, yang diterapkan hanya untuk solusi terbaik dengan probabilitas yang kecil.

*Cold restart* adalah proses penggantian populasi, dimana semua kromosom digantikan sekaligus oleh N kromosom baru. Ada dua masalah yang menyebabkan penggantian populasi perlu dilakukan. Pertama, rendahnya variasi atau keanekaragaman pada populasi yang dapat menyebabkan AM konvergensi prematur ke solusi yang tidak cukup baik.. Dan kedua, ketika AM telah mencapai stagnansi, dimana tidak ada lagi peningkatan berarti setelah generasi tertentu, namun mungkin saja populasinya masih cukup bervariasi, sehingga seharusnya bisa menemukan solusi yang baik.

Proses *cold restart* adalah setiap kali *makespan* terkecil dari populasi tidak berubah setelah lebih dari $C_r$ generasi, *cold restart* dilakukan, dengan prosedur sebagai berikut:

- Tiga solusi terbaik tetap dipertahankan pada populasi baru.
- 40 % dari populasi dihasilkan dari hasil mutasi terhadap tiga solusi terbaik di atas.
- Sisanya dibentuk secara random.

Berikut adalah *Pseudocode* metode AM dan GRASP untuk menyelesaikan

PFSP:
*Begin*
Tentukan parameter GRASP dan Algoritma Memetika
Baca data (waktu proses setiap pekerjaan pada setiap mesin);

********* GRASP *********
Terapkan metode NEH;
Masukkan solusi NEH ke *Pool*;
Terapkan *Local Search* pada solusi NEH;
Masukkan solusi *Local Search* ke *Pool*;
*CurrSolution = BestSolution*;
*for* iter= 1 sampai dengan maxIter do
Terapkan ILS ;
Masukkan solusi ILS ke Pool (jika memenuhi kriteria);
Terapkan PR (jika memenuhi kriteria);
Masukkan solusi PR ke Pool (jika memenuhi kriteria);
*Update CurrSolution* jika solusi PR lebih baik
*Update CurrSolution* jika solusi ILS lebih baik
*Restart Pool* (jika tidak ada peningkatan *BestSolution* setelah sejumlah iterasi tertentu)
*endfor*;
*********Algoritma Memetika*********
Bentuk populasi awal sesuai ukuran *popsize*;
Terapkan *Local Search* untuk semua solusi dalam populasi awal
for gen = 1 sampai dengan maxGen do
Hitung nilai *fitness* tiap kromosom pada populasi;
Bentuk populasi baru dengan seleksi;
Terapkan *crosssover* dan mutasi (jika memenuhi kriteria);
Terapkan *Local Search* (jika memenuhi kriteria);
Terapkan *cold restart scheme* (jika memenuhi kriteria);
*endfor;*
*end;*

## HASIL DAN PEMBAHASAN

Metode AM dan GRASP untuk menyelesaikan PFSP diimplementasikan dalam sebuah program dengan menggunakan MATLAB 7. Program dijalankan pada *Personal Computer* (PC) dengan prosesor Intel Celeron 2.26GHz, memori 256 Mb, dan sistem operasi *Microsoft Windows XP Professional version 2002* .

Berikut diberikan beberapa hasil percobaan untuk melihat kinerja AM dan GRASP dalam menyelesaikan masalah PFSP dengan ukuran yang bervariasi. Masalah penguji yang digunakan adalah 12 data masalah pengujian dengan ukuran yang bervariasi, yaitu 20 x 5, 20 x 10, 20 x 20 dan 50 x 5, 50 x 10, dan 100 x 5 diambil dari *Taillard's Benchmark*.

Percobaan AM dan GRASP berikut ini menggunakan nilai parameter sebagai berikut: Iterasi maksimum GRASP: 50, PRF: 2, generasi maksimum:50 atau 100, Ukuran populasi: 20, probabilitas mutasi: 0.01, probabilitas PX: 0.5, probabilitas LS: 0.05, dan Cr: 20. Hasil percobaan ditampilkan pada tabel di bawah ini.

Tabel memperlihatkan bahwa *error* relatif yang dihasilkan tidak lebih dari 2 %. Dari 10 masalah yang diuji, 8 di antaranya memiliki *error* relatif yang kurang dari 1 % dan hanya 4 masalah yang memiliki *error* relatif antara 1% - 2%. Dapat dilihat juga bahwa untuk mencapai *error* relatif tersebut, waktu komputasi rata-rata cenderung meningkat seiring peningkatan ukuran data. disimpulkan bahwa metode AM dan GRASP cukup baik untuk menyelesaikan PFSP. Metode AM dan GRASP menghasilkan jadual dengan *makespan* yang memiliki *error* relatif tidak lebih dari 2%.

Berdasarkan hasil perbandingan

**Tabel Nilai *Makespan* dan Waktu Komputasi dari Hasil Percobaan AM dan GRASP untuk Menyelesaikan PFSP**

| Nama Data | Ukuran | *Makespan*(atas) dan waktu komputasi ( *second*)(bawah) dari percobaan ke- | | | | | | | | | | *Makespan* terbaik(atas) Waktu rata-rata(bawah) | BKS | *error* relatif |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | | |
| ta03 | 20 x 5 | 1081 | 1080 | 1081 | 1088 | 1085 | 1088 | 1088 | 1091 | 1081 | 1084 | 1081 | 1081 | 0.0033 |
| | | 6.6 | 4.7 | 5.9 | 6.1 | 6.8 | 7.4 | 7.2 | 7.1 | 6.4 | 5.1 | 6.33 | | |
| ta08 | 20 x 5 | 1211 | 1206 | 1206 | 1206 | 1206 | 1206 | 1206 | 1211 | 1212 | 1206 | 1206 | 1206 | 0.0013 |
| | | 8.6 | 7.6 | 7.3 | 7.9 | 6.5 | 7.7 | 6.7 | 8.3 | 8.1 | 7.1 | 7.6 | | |
| ta69 | 20 x 5 | 1230 | 1235 | 1236 | 1230 | 1236 | 1236 | 1238 | 1238 | 1239 | 1239 | 1230 | 1230 | 0.0046 |
| | | 7.9 | 9.2 | 10.4 | 9.5 | 7.5 | 6.8 | 6.8 | 9.1 | 7.3 | 8.1 | 8.26 | | |
| ta10 | 20 x 5 | 1108 | 1108 | 1112 | 1108 | 1111 | 1109 | 1111 | 1113 | 1111 | 1112 | 1108 | 1108 | 0.0021 |
| | | 10 | 9.2 | 8.3 | 8.5 | 7.5 | 6.3 | 7.4 | 6.6 | 7.4 | 10.3 | 8.15 | | |
| ta12 | 20 x 10 | 1686 | 1684 | 1691 | 1691 | 1693 | 1691 | 1695 | 1694 | 1689 | 1695 | 1684 | 1659 | 0.0192 |
| | | 9 | 8.9 | 12.1 | 9.3 | 12.3 | 12.1 | 9.4 | 10.2 | 9.4 | 12.8 | 10.57 | | |
| ta13 | 20 x 10 | 1513 | 1513 | 1519 | 1518 | 1517 | 1518 | 1509 | 1514 | 1518 | 1513 | 1509 | 1496 | 0.0128 |
| | | 9.4 | 10 | 9.8 | 10.9 | 13.6 | 11.4 | 11.4 | 10.2 | 7.8 | 8.1 | 10.26 | | |
| ta22 | 20 x 20 | 2132 | 2122 | 2120 | 2128 | 2124 | 2126 | 2120 | 2120 | 2120 | 2120 | 2120 | 2099 | 0.0115 |
| | | 9.1 | 10.4 | 11.8 | 7.1 | 6.3 | 9.6 | 9 | 9.3 | 10 | 10.3 | 9.29 | | |
| ta26 | 20 x 20 | 2245 | 2230 | 2232 | 2230 | 2245 | 2249 | 2243 | 2244 | 2244 | 2234 | 2243 | 2226 | 0.0097 |
| | | 10.2 | 9.1 | 9.9 | 7.1 | 5.9 | 12.1 | 8.6 | 10.7 | 10.6 | 11.2 | 9.51 | | |
| ta31 | 50 x 5 | 2729 | 2724 | 2729 | 2724 | 2729 | 2729 | 2724 | 2724 | 2729 | 2724 | 2724 | 2724 | 0.0009 |
| | | 90.4 | 20.8 | 23.8 | 10.4 | 90.2 | 92.2 | 20.8 | 22.1 | 20.4 | 90.9 | 50 | | |
| ta33 | 50 x 5 | 2621 | 2667 | 2631 | 2621 | 2623 | 2627 | 2623 | 2621 | 2623 | 2622 | 2621 | 2621 | 0.0012 |
| | | 19.7 | 25.8 | 24.6 | 20.6 | 26 | 29.6 | 21.3 | 68 | 28.2 | 23.6 | 34.68 | | |
| ta41 | 50 x 10 | 3123 | 3123 | 3097 | 3103 | 3119 | 3108 | 3104 | 3102 | 3105 | 3105 | 3097 | 3063 | 0.0149 |
| | | 82.7 | 80.3 | 76.5 | 60.4 | 174.6 | 95 | 75.9 | 69 | 176.6 | 61.7 | 94.66 | | |
| ta93 | 100 x 5 | 3206 | 3213 | 3213 | 3213 | 3213 | 3193 | 3193 | 3205 | 3213 | 3205 | 3193 | 3173 | 0.0061 |
| | | 283.8 | 297.4 | 231.7 | 181.2 | 145.6 | 269.3 | 221.5 | 205.2 | 266.4 | 195.9 | 229.8 | | |

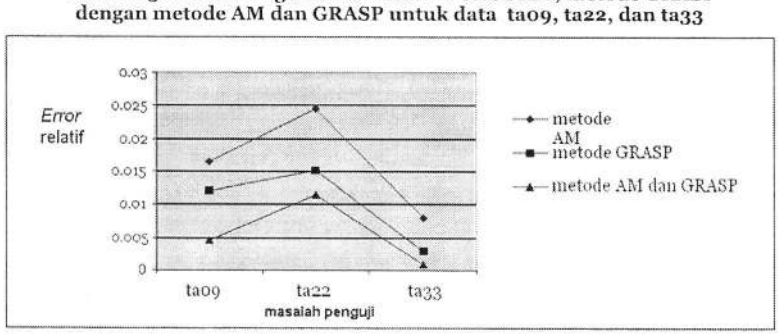
Di bawah ini disajikan beberapa hasil percobaan yang akan digunakan untuk melihat perbandingan kinerja metode AM dan metode GRASP dengan kombinasi AM dan GRASP. Di sini dilakukan percobaan metode AM dan metode GRASP untuk 3 masalah penguji, ta09, ta22, dan ta33, dengan kriteria berhentinya adalah CPU *time* dengan besar yang sama dengan rata-rata waktu komputasi yang diperoleh metode AM dan GRASP pada tabel di atas untuk setiap masalah tersebut. Hasil percobaan ditampilkan pada Gambar 3.

Dari hasil percobaan dapat dilihat bahwa rata-rata *error* metode AM dan GRASP lebih baik dibanding metode AM dan metode GRASP sendiri-sendiri. Rata-rata *error* metode AM dan metode GRASP sebenarnya masih bisa dikecilkan lagi, jika waktu komputasinya ditambah. Hal ini berarti metode AM dan GRASP lebih cepat konvergen ke solusi optimal dibandingkan dengan metode AM dan metode GRASP sendiri-sendiri.

## KESIMPULAN

Berdasarkan hasil percobaan yang telah dilakukan pada beberapa data masalah penguji dari *Taillard's Benchmark*, didapatkan bahwa kombinasi AM dan GRASP lebih cepat konvergen ke solusi optimal dibandingkan metode GRASP atau Algoritma Memetika. Maka dapat disimpulkan, kombinasi metode AM dan GRASP menghasilkan kinerja yang lebih baik dibanding metode AM dan metode GRASP sendiri-sendiri.

**Gambar 3. Perbandingan *error* relatif metode AM, metode GRASP dengan metode AM dan GRASP untuk data ta09, ta22, dan ta33**




## DAFTAR PUSTAKA


Aiex, R.M., Binato,S., Resende, M.G.C. 2001. Parallel GRASP with Path Relinking for Job Shop Scheduling, *AT&T Labs Research Technical Report, USA*. To appear in Parallel Computing.

El-Bouri, Ahmed. "*A Hybrid Genetic Algorithm for Flowshop Scheduling.*" Retrieved 2008 from http://www.Umoncton.ca/cie/conferences/29thconf/29thICCIE/Papers/paper04.pdf

Digaspero, L. 2003. *Local Search Techniques for Scheduling Problems:*

*Algorithms and Software Tools*. Ph.D. Thesis 1061-1071.

Yamada, T., Reeves, C.1997. *Permutation Flowshop, Scheduling by Genetic Local Search*. IEE conference publication, 232-238, IEE

Murata, T., Ishibuchi, H. & Tanaka, H. 1996. *Genetic algorithms for flowshop scheduling problems*, Computers Ind. Engng., 30(4).

Obitko, M. 1998. *Introduction to Genetic Algorithm*. http://www.obitko.com/tutorials/genetic-algorithms/

Ravetti,M.G Nakamura, F.G. Meneses,C.N. Resende,M.G.C. Mateus, and G.R.Pardalos,P. 2006. *Hybrid Heuristics for the Permutation Flow Shop Problem*. AT&T Labs Research Technical Report TD-6V9MEV, Shannon Laboratory, Florham Park, NJ 07932 USA.

Ruiz, R., Maroto, C., Alcaraz, J. 2006. Two new robust genetic algorithms for the flowshop scheduling problem. *OMEGA, The International Journal of Management Science* 34, 461–476.

Reeves,C. R., Yamada, T. 1998. Genetic Algorithms, Path Relinking and the Flowshop Sequencing Problem. *Evolutionary Computation Journal* (MIT press), Vol.6 No.1, pp. 230

Šeda, Miloš. 2007. Mathematical Models Of Flow Shop And Job Shop Scheduling Problems. *Proceedings Of World Academy Of Science, Engineering And Technology* Volume 25 November 2007 Issn 1307-6884

Stutzle, Thomas. 1998. *Applying Iterated Local Search to the Flow Shop Problem*. Technical Report, AIDA-98-04, Darmstad University of Technology, Computer Science Department, Intellctics Group, Darmstad, Germany